A PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

III, NÚMERO 41

975 Ptas.



GESTIÓN DE MEMORIA DINÁMICA CON C

Descubre sus secretos

MERCED

El futuro de Intel

JASMINE

Bases de datos orientadas a objetos

VISUAL C++

Mapas de mensajes

REDES LOCALES

El protocolo TCP/IP

VISUAL CAFÉ

La última apuesta de Symantec

PERL

Curso de programación

CONTENIDO DEL CD-ROM

- DB2 Universal Database
- Borland DataGateway for Java
- Mirror del CPAN
- Star Office 4.0 para Linux
- Service Pack 3 del Visual Basic 5.0
- GetRight 3.02
- Microsoft Project 98 Trial
- Fuentes de los artículos de la revista
- Y mucho más...

8 413042 303299

SÓLO PROGRAMADORES

Número 41
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Director Adjunto
Enrique Maldonado
enriquem@towercom.es
Colaboradores
Enrique Díaz, Xesco Hernández, Ernesto Schmitz, José Eulalio Poza, Chema Álvarez, Manuel Herrán Gascón, Jorge Figueroa, Juan José Taboada.

Maquetación
Susana Llano
Tratamiento de Imagen
María Arce Giménez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Isabel Bojo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Laboratorio
Óscar Rodríguez (jefe)
Javier Amado
Servicio Técnico
Óscar Casado

Filmación
Megatipo
Impresión
Gráficas Reunidas
Distribución
SGEL
Distribución en Argentina
Capital: Huesca y Sanabria
Interior: D.G.P.

TOWER COMMUNICATIONS
Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
carmina@towercom.es

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN

EDITORIAL

Tiempos de cambio

Hace sólo unos pocos años terminé mis estudios universitarios y parece que hoy en día todos los lenguajes y técnicas de programación, que me enseñaron por aquellos tiempos, están hoy obsoletos. Lenguajes tan famosos, como el mismísimo Cobol, han quedado relegados a un segundo plano, salvo en los grandes sistemas donde un efecto "dinosaurio" evita que caigan y sean afectados por nuevas tendencias como el popular Java.

Muchas veces escuchamos cómo los usuarios del mundo del PC se quejan porque la tecnología y las prestaciones de los ordenadores avanzan tan rápidamente, que tienen que estar adquiriendo nuevos equipos cada dos años si quieren poder ejecutar las últimas versiones de los programas más populares, pero, ¿es que nadie se acuerda de los programadores? El que lleve al menos un par de años programando, se dará cuenta que es necesario estar continuamente reciclando conocimientos o al cabo de unos meses estará programando en un lenguaje o versión prehistórica.

La prensa especializada ayuda al programador a comprender cuales son las nuevas técnicas de programación, pero el paso final de aprender siempre le tocará al programador profesional, que puede que no tenga el tiempo suficiente. Las últimas versiones de los lenguajes de programación más populares han ayudado mucho a acelerar la creación de programas, como es el caso de los lenguajes visuales, que ahorran al programador la creación de código para las tareas más comunes.

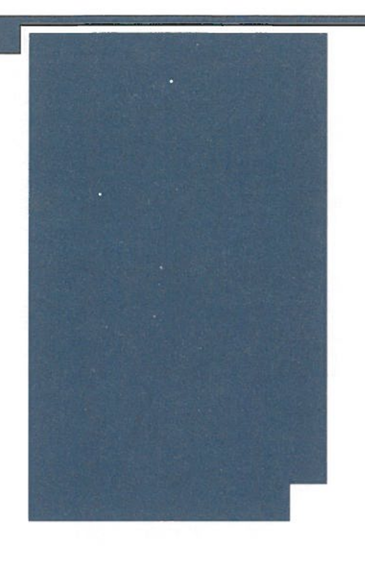
Puede que dentro de muchos años el software se cree a medida en máquinas de veinte duros, pero seguramente antes, hasta los expertos habrán quedado aplastados por la gran avalancha de versiones, lenguajes nuevos y estándares, que van naciendo día a día.

La solución es difícil pero… ¿no podría crearse una organización libre que crease un lenguaje único para todos los sistemas? Es un sueño, debido a las grandes incompatibilidades que existen y a los diferentes requerimientos de cada sistema, pero tal vez algún día el árbol de la evolución de la programación deje de tener tantas ramas y crezca a lo alto. ¿Es un sueño? Quizás.

Tenéis a vuestra disposición la dirección de correo electrónico de la revista: solop@towercom.es, donde podéis enviar vuestras sugerencias sobre los contenidos, los temas que queráis que incluyamos y los productos que os gustaría que distribuyéramos con el CD-ROM, que todos los meses regalamos. Sois nuestros mejores colaboradores.

SÓLO PROGRAMADORES





Noticias

LA INTERNATIONAL STANDARS ORGANIZATION APRUEBA EL PROCESO DE ESTANDARIZACIÓN DE JAVA

La Organización Internacional para la Estandarización (International Organization for Standardization, ISO) ha aprobado por mayoría el inicio de los procedimientos para la estandarización de las especificaciones de Java de Sun Microsystems. A partir de esta decisión, Sun debe presentar las especificaciones de la plataforma Java para conseguir su estandarización.

La International Organization for Standartization es una federación mundial no gubernamental de organizmos de estandarización nacionales de más de 100 países. Las deliberaciones que ISO revisa resultan Estándares Internacionales para múltiples actividades industriales. La estandarización Internacional es voluntaria, basada en consenso y está dirigida al mercado.

La plataforma Java es ya un estándar de facto a nivel mundial para la informática a través de Internet. Los estándares de facto son importantes y de hecho actualmente la industria confía plenamente en ellos. La compañía Sun, sin embargo, sostiene que se sirve mejor a la industria si una tecnología de esta magnitud está validada por una organización de estandarización internacional de la talla de ISO/IEC. De este modo, tanto gobiernos como universidades y otras instituciones que precisan el uso de tecnologías estándar podrán disponer mejor de soluciones basadas en Java para su explotación. Sun cree que esta actitud es la que debe tomar toda la industria informática en general.

Tras esta aceptación como PAS Submitter (*Publicly Avaible Specification*: Especificación Disponible Públicamente), Sun ahora tiene el permiso para presentar las especificaciones sobre las que subyacen las tecnologías Java para su consideración como estándar internacional. Sun tiene un plazo de hasta dos años para hacer esta presentaciones.

Sun presentará las especificaciones para la plataforma Java puesto que es necesario para soportar "Write Once Run Anywhere". El conjunto de las especificaciones para la Virtual Machine de Java, del lenguaje Java y del núcleo de las librerías de clases conforman la especificación de la plataforma Java.

El modo en el que Sun manejará Java no cambiará. El proceso de la empresa para hacer evolucionar la plataforma es y seguirá siendo un proceso de colaboración guiado por la industria. Sun continuará trabajando con los líderes de la industria para definir especificaciones para las tecnologías Java. Sun ha afirmado que la estabdarización no ralentizará el desarrollo de la plataforma Java.

En la votación final, 20 países respondieron afirmativamente, dos negativamente y hubo dos abstenciones.

COMPAQ PRESENTA SUS PRODUCTOS PARA EL MUNDO GRÁFICO

Compaq presentó en su Primer Festival Gráfico sus soluciones para el diseño, creación y tratamiento de imágenes sobre plataformas multiproceso con sus estaciones de trabajo 5100, 6000 y 8000.

Las estaciones de trabajo de Compaq están diseñadas para las aplicaciones que requieren elevados niveles de rendimiento en el procesamiento y en los gráficos, como pueden ser aplicaciones CAD (Diseño Asistido por Ordenador), CAE (Ingeniería Asistida por Ordenador), DCC (Creación de Contenidos Digitales) como animación 3D, edición de vídeo y diseño de webs, y complejos modelos financieros.

La Compaq Professional Workstation 5100, basada en la arquitectura de sistemas paralelos de Compaq, ofrece un rendimiento sin precedentes con procesadores Pentium II de Intel y Microsoft windows NT, en un diseño de ordenador de sobremesa de espacio reducido y de precio asequible.

La estación de trabajo de Compaq 6000 ofrece la potencia de una estación de trabajo en un diseño de torre y orientado a las aplicaciones de altas prestaciones como diseño mecánico, diseño y animación 2D/3D, edición de vídeo, EDA (Electronics Design Automation, Diseño Electrónico automático). Admite hasta dos Pentiums a 266 MHz o 300 MHz, con un caché de 512 Mb de memoria.

La estación de trabajo 800 proporciona las capacidades avanzadas de multiproceso para la mayoría de las aplicaciones intensivas en recursos, como el análisis de elementos finitos, transformación final de complejas animaciones 3D, contenido multimedia y operaciones EDA como diseño de circuitos integrados.

Las estaciones de trabajo de Compaq 6000 y 8000 están diseñadas para las aplicaciones que requieren elevados niveles de rendimiento en capacidad de proceso y en gráficos, como pueden ser aplicaciones CAD (Compter Aided Design, Diseño Asistido por Ordenador), CAE (Computer Aided Engineering, Ingeniería Asistida por Ordenador), etc.

UN PASO MÁS EN EL PROYECTO JAVA CARD 2 DE SUN

La especificación 2.0 de la API (Interfaz de programación de aplicaciones) Java Card ya se encuentra finalizada y disponible en la siguiente dirección: www.java.sun.com/products/javacard.

Java Card 2.0 es un proyecto original para la creación de aplicaciones que se ejecuten en tarjetas inteligentes. Una smarcard es una tarjeta de plástico que tiene un aspecto prácticamente igual al de una tarjeta de crédito. La diferencia es que una smartcard tiene un chip informático que almacena y procesa información.

Las tarjetas que integran la tecnología Java Card 2.0 se utilizarán para almacenar y actualizar datos de información sobre cuentas, dinero y personales.

Estas smartcards abarcarán una amplia variedad de sectores, incluidos los servicios financieros, telefonía sanidad, acceso a Internet y comercio electrónico.

Breves

WINDOWS NT CUATRIPLICA SUS VENTAS EN ESPAÑA

Microsoft ha anunciado que las ventas de licencias de Windows NT Workstation han sobrepasado los 11 millones de unidades en todo el mundo, mientras que en España el número de licencias se ha multiplicado por cuatro. El 70% de las organizaciones de tamaño medio están implantando este sistema operativo o están considerando la posibilidad de hacerlo, mientras que el 25 % de las pequeñas empresas están trabajando en entornos mixtos basados en Windows NT Workstation y en Sindows 95. Al mismo tiempo, un buen número de compañías de renombre han adoptado el sistema operativo cliente de Microsoft como estándar de su negocio.

UN 95% DE DESCUENTOS PARA ESTUDIANTES EN PRODUCTOS ORACLE

Por el dinero que un joven se gasta en un fin de semana, alrededor de 5.000 ptas., según Oracle, esta empresa ofrece a los universitarios españoles la posibilidad de adquirir la licencia de su base de datos y de las herramientas de desarrollo.

Esta iniciativa, denominado Licencia Oracle para Estudiantes, está pensada para ofrecer a los universitarios de nuestro país la formación en tecnología Oracle, una de las más demandadas en el mercado laboral. Con este iniciativa los estudiante pueden obtener por un precio de 5.000 ptas., un CD-ROM con la base de datos Personal Oracle7 y las herramientas de desarrollo del producto OracleDeveloper/2000. Este precio supone un descuento del 95% con respecto al precio de estos productos en el mercado.

JBUILDER CLIENT/SERVER
Borland lanza su nueva herramienta Java de alto nivel

Borland International ha anunciado el lanzamiento y disponibilidad inmediata de la *suite* JBuilder Client/Server: una incorporación de alto nivel a la línea de herramientas de desarrollo visual Java de la compañía.

Borland Jbuilder es una familia de herramientas de desarrollo visual altamente productivas para crear aplicaciones independientes de la plataforma de alto rendimiento utilizando el lenguaje de programación Java. El entorno dimensionable basado en componentes de JBuilder está diseñado para todos los niveles de proyectos de desarrollo de "Redes de información" que van desde *applets* y aplicaciones que requieren conectividad basada en redes hasta sistemas multinivel distribuidos, de empresa y cliente/servidor.

La familia de productos JBuilder ofrece la creación de componentes JavaBeans más rápida y fácil del mercado, una arquitectura de base de datos dimensionable y la capacidad de producir aplicaciones, applets y JavaBeans independientes de la plataforma "100% Pure Java". El entorno abierto del producto soporta 100% Pure Java, JDK 1.1, JavaBeans, JFC, CORBA, RMI, JDBC y todos los servidores de base de datos corporativos principales.

JBuilder Client/Server soporta el desarrollo corporativo de aplicaciones Java multiplataforma a gran escala integrando VisiBroker ORB de Visigenic para integración CORBA transparente, Borland DataGateway Enterprise para conectividad de base de datos de alto rendimiento, software InterSolv Pvcs para gestión avanzada de equipos de desarrollo y control de versiones y las probadas herramientas SQL de Borland para el desarrollo robusto de esquemas cliente/servidor.

BORLAND ABSORVE A LA EMPRESA VISIGENIC SOFTWARE

Borland Internacional, un importante proveedor de tecnologías de desarrollo de aplicaciones cliente/servidor y multinivel, ha adquirido recientemente la empresa Visigenic Software. Esta última empresa es proveedor líder de tecnología CORBA Object para el mercado de informática.

La adquisición propuesta de Borland está destinada a establecer a la compañía como líder en una de las tendencias más significativas que impactan en la informática actual: la informática distribuida de empresa.

La conectividad general de Internet y la disponibilidad instantánea de datos han creado una demanda sin precedentes en las organizaciones IT para crear nuevas aplicaciones e infraestructura que proporcione información en tiempo real a los empleados, clientes y proveedores de una manera significativa. Para satisfacer esta demanda, los gestores IR corporativos deben encontrar un modo de crear aplicaciones nuevas y fáciles de utilizar que manejen sin problemas datos almacenados en una multitud de plataformas de la empresa. Muchas empresas han reconocido ya que deben superar la monolítica arquitectura de los minfrmae y la capacidad de dimensión limitada de la arquitectura cliente/servidor de dos niveles para alcanzar una arquitectura de informática distribuida y más flexible para la empresa. Un modelo de informática distribuida bgasado en componentes permite a las organizaciones IT crear una infraestructura que se adapte al cambio constante y responda a las oportunidades del mercado.

La adquisición de Visigenic, según el presidente de Borland, Del Yocam, está destinada a acelerar el crecimiento de la compañía como proveedor de tecnología fundamental para el mercado de grandes empresas. La incorporación de las tecnologías distribuidas de objetos pioneras de Visigenic permitirán a los clientes de Borland combinar aplicaciones de legado, cliente/servidor, Internet y basadas en objetos en una única aplicación de Red de Información distribuida. Las aplicaciones de Red de Información permiten a las organizaciones dar a sus empleados, clientes y proveedores acceso en tiempo real a información significativa de fuentes de datos corporativas de la empresa. La combinación de tecnologías Borland y Visigenic permitirán a los clientes de Borland desarrollar, desplegar y gestionar sofisticadas aplicaciones distribuidas en múltiples plataformas, incluyendo Windows, Unix, AS/4000 y MSV.

Según el acuerdo firmado, los accionista de Visigenic recibirán 0,81988 de acción del capital común de Borland por cada acción del capital común de Visigenic. Aproximadamente 12 millones de acciones del capital común de Borland serán entregadas al cierre de la transacción, que se estima ocurrirá el primer trimestre de 1998.

El consejo de directores de ambas compañías ha aprobado la fusión, que está sujeta a conformidad por parte de los accionistas de ambas empresas, así como a las habituales aprobaciones reguladoras.

Windows NT 4.0 Server, Guía Avanzada

La editorial Prentice Hall sigue con su rentable labor de editar guías y manuales aclaratorios sobre todo el software (sistemas operativos, ect.) que tiene más aceptación en el mercado informático.

En este caso, le ha tocado el turno al sistema operativo Windows NT. en su versión 4.0. Esta guía tiene como objetivo principal introducir al lector en las dos versiones existentes de windows NT (la versión Server y la WorkStation), tratando conjuntamente las muchas características que poseen en común.

La guía se estructura en dos partes, claramente diferenciadas. La primera parte del libro se basa en el desarrollo de una red sencilla con un único servidor y con un controlador de dominio. Este capítulo sirve para introducir los conceptos básicos que todo buen administrador debe conocer.

Dentro de la primera parte se trata cómo proceder a la primera instalación de Windows NT creando el dominio y tratando las herramientas para la administración de usuarios, elección del sistema de ficheros a utilizar en el servidor y estaciones de trabajo, asignación de permisos y derechos a los usuarios para delimitar las operaciones que podrán realizar, así como la configuración de la red, de los servidores de impresión y una administración básica de los recursos compartidos en la red.



La segunda parte profundiza en todos aquellos elementos necesarios para implementar soluciones de redes más complejas. Se entra en detalle en las opciones de instalación y cómo se administran los dominios distintos en la empresa.

Otro factor importante es la configuración de los usuarios y todas las capacidades de seguridad física incorporadas por Windows NT. También se ha creído conveniente incluir un capítulo para ayudarnos en la problemática de medir el rendimiento de los servidores.

Un tema importante de esta segunda parte es todo lo referente a la conectividad de Windows NT con otras redes tales como NetWare de Novel, TCP/IP, el mundo MAC e, incluso, la comunicación vía módem u otro vínculo WAN. También se introduce la configuración de los servidores de Internet Information Server para instalar servidores de Internet.

Tras estas dos partes, se incluye un interesante apéndice con información relativa al *browser* (servicio de Examinador de red), a las utilidades TCP/IP y a la arquitectura de Windows NT.

Editorial: Prentice Hall
Número de páginas: 496
PVP: 4.317 Ptas. + IVA

Autores: José Manuel Martínez Delgado y Xavier Martínez Balart
Idioma: castellano
Nivel: avanzado

Nace Jasmine, la primera base de datos orientada a objetos

Inma Portalo

Computer Associates International Inc (CA) y Fujitsu Limited anunciaron en Madrid durante el mes de diciembre la disponibilidad mundial de Jasmine, la primera solución orientada a objetos para construir la próxima generación de sistemas de negocio en Internet y los entornos de informática cliente/servidor.

Jasmine ofrece a los desarrolladores una plataforma orientada a objetos ideal para las aplicaciones multimedia dinámicas, además de proporcionar la integridad y gestión práctica de datos necesarias para las soluciones empresariales.

La arquitectura orientada a objetos de Jasmine elimina los gastos generales asociados con el mapeo de datos para las estructuras relacionales. Utilizando un enfoque orientado a objetos de forma inherente en combinación con las librerías SQL para llegar a los repositorios de datos existentes, los usuarios de Jasmine aprovechan los beneficios de la tecnología orientada a objetos sin necesidad de sacrificar las inversiones realizadas en datos relacionales y aplicaciones.

JADE, Entorno de Desarrollo de Aplicaciones Jasmine, proporciona un entorno de desarrollo de aplicaciones fácil de utilizar y multimedia con herramientas para el diseño rápido de aplicaciones de depuración sofisticadas. La arquitectura abierta de Jasmine también permite a los desarrolladores utilizar una gran variedad de herramientas populares, incluyendo Java cien por cien, C, C++, HTML nativo y cualquier otra herramienta ActiveX, Jasmine también facilita el desarrollo de librerías funcionales y especializadas que contemplan las actuales y futuras necesidades de negocio.

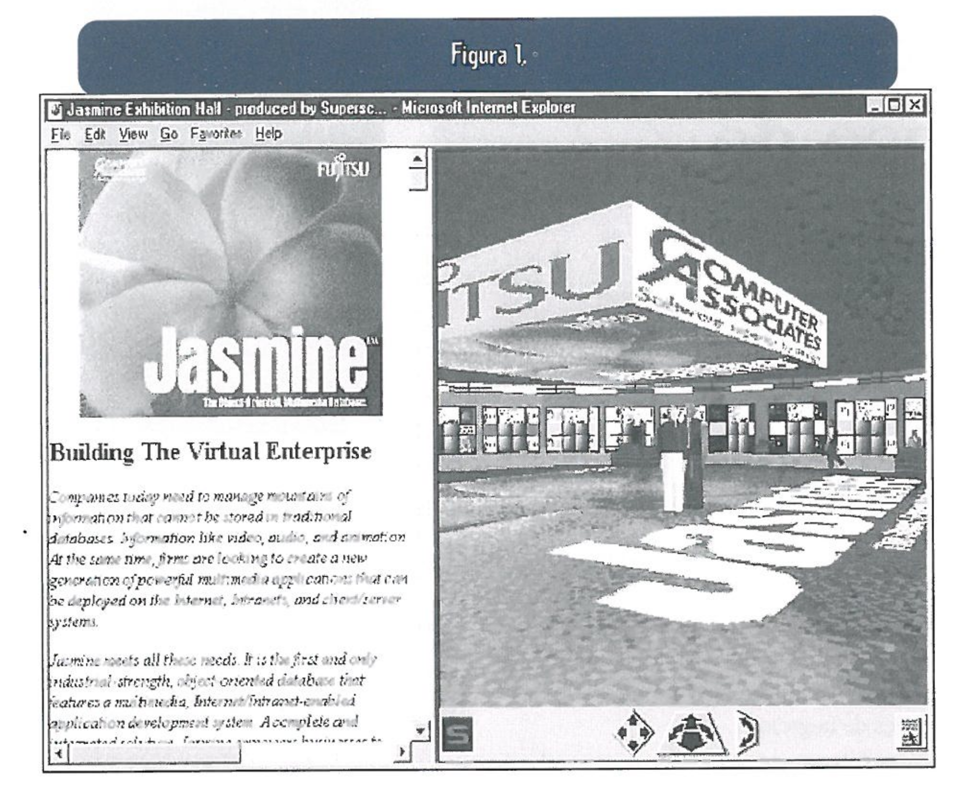
Jasmine proporciona acceso nativo a fuentes relacionales de datos y de otras clases, entre las que destacan OpenIngres, Oracle, Informix, SQL Server, CA-IDMS, CA-Datacom y DB2. El soporte para el estándar OLE DB y ODBC permite a Jasmine trabajar con los generadores de informes más conocidos y herramientas de soporte a la toma de decisiones.

Su completo y abierto acceso permite a las compañías aprovechar las inversiones existentes, mientras que se construye la próxima generación de aplicaciones de negocio. Además de los cientos de usuarios beta, consultores y VARs que ya están utilizando Jasmine, docenas de fabricantes de hardware y software han aprovechado las licencias gratuitas de desarrollo Jasmine, para construir soluciones complementarias.

Si algo caracteriza a esta nueva herramienta es su sencillez tanto para programadores como para no programadores. Esta facilidad de implementación junto con el tremendo poder de Internet (o intranets corporativas y extranets), permiten a Jasmine disminuir los costes derivados de encontrar compradores y

Gracias a la base de datos Jasmine se pone al alcance de todos los programadores la posibilidad de utilizar Java, C, C++, HTML nativo y cualquier herramienta ActiveX

Figura 1.



vendedores, ampliar mercados geométricamente, centrarse en la producción y los procesos de pago.

Las aplicaciones de Jasmine se pueden implementar en cualquier entorno informático, desde Internet hasta las intranets corporativas heterogéneas a las redes privadas de valor añadido, y puede operar en browsers Web, a través de Microsoft Internet Explorer y los plugs-in Netscape.

Las librerías integradas de CA y otros fabricantes ayudan a los desarrolladores de Jasmine a construir e implementar aplicaciones, de manera rápida y eficaz. Esto es importante no sólo para los programadores profesionales, sino también para los casuales que cada vez más se encuentran involucrados en esta tecnología corporativa, incluyendo webmeasters, analistas de la industria y otros usuarios sofisticados.

Jazmine integra sencillas librerías para la creación y gestión de datos multimedia, incluyendo mapas de bits, animación, características audio y vídeo. Soporta una extensa gama de aplicaciones de nueva generación, desde comercio electrónico y soporte al usuario para aplicaciones industriales específicas en sectores como seguros, servicios financieros, salud y telecomunicaciones. La conectividad de datos empresariales y la adhesión a los estándares industriales, convierten en tarea sencilla la conexión de las aplicaciones Jasmine al resto de la información empresarial.

La versión inicial de Jasmine, disponible ya en el mercado, soportará servidores UNIX y Windows NT, además de usuarios de Windows 95 y Windows NT. Su precio es de 800 dólares.

Charles B. Wang, presidente de Computer Associates ha denunciado el hecho de que, durante años, la tecnología de objetos se ha quedado en construir muchas promesas y pocas realidades. Esto ha llevado a que los programadores hayan visto obligados a aprender lenguajes complejos y a aunar bases de datos y herramientas no integradas. Éste, según Wang, ha sido el motivo de que su empresa (CA) haya formado un equipo con Fujitsu, desarrollador de tecnología orientada a objetos. El objetivo de ambas compañías ha sido el de desarrollar y comercializar a nivel internacional esta innovadora tecnología.

Durante la inevitable oleada de reconocimientos y agradecimientos entre ambas empresas, Tatsumi Furukawa, director del Consejo de Dirección y presidente del grupo de negocio multimedia de Fujitsu, ha reconocido la importancia de la experiencia adicional y los recursos de un líder informático como CA para llevar a cabo el proyecto. Jasmine ofrece a las compañías el poder del trabajo con una gran variedad de tipos de objetos, entre los que se incluyen gráficos, animación, audio y vídeo, para crear nuevas aplicaciones orientadas al mercado global.

Del éxito de la base de datos Jasmine responden ya varias empresas que ha trabajado con ella.

Kevin Smith, director de sistemas accesorios y componentes de Toyota asegura que Jasmine y CA fueron decisivas en la tarea de ayudar a Toyota a explorar las oportunidades para utilizar aplicaciones multimedia e incrementar las ventas de accesorios auténticos. Smith sostiene que su empresa quería crear una aplicación de distribuidores que atrajera por completo la atención de los usuarios. Una base de datos orientada a objetos era esencial para lo que se pretendía construir y no había tiempo para operar con lenguajes de programación complicados como C++. Se necesitaba un entorno de desarrollo visual e intuitivo que se integrara de manera transparente con la base de datos.

Por su parte, Michel Barry, director técnico de L'Oreal, reconoce que Jasmine proporciona la solución óptima para centralizar y gestionar información compleja. Y, de acuerdo con su capacidad web, ofrece una base sólida para un proyecto intranet.

Barry asegura que la architectura orientada a objetos de Jazmine es muy atractiva para su empresa. Los almacenes de bases de datos vídeo como vídeo, audio como audio, imágenes como imágenes...

El protocolo TCP/IP

Enrique Díaz Trobo

Continuamos asentando las bases de nuestra futura intranet. En esta ocasión veremos el protocolo TCP/IP. Tras estas siglas se esconden multitud de protocolos y utilidades. También introduciremos el concepto de intranet y extranet. Dos términos que constituyen un futuro que ya está aquí.

El protocolo TCP/IP fue desarrollado en 1972 por el Departamento de Defensa de Estados Unidos. Pensado en principio para su integración en la red de área extensa ARPANET del Departamento de Defensa, fue adoptada rápidamente en Internet para darle un uso mucho menos belicoso.

Su nombre proviene de dos de los protocolos más importantes de la familia de protocolos de Internet; el *Transmission Control Protocol* (TCP) y el *Internet Protocol* (IP). Además de estos protocolos hay alguno más que se esconde tras estas siglas, como son el Protocolo de datagramas de usuario (UDP), el Protocolo de resolución de direcciones (ARP) y el Protocolo de mensajes de control de Internet (ICMP).

En los protocolos TCP/IP se conceptúa la red en un modelo de cuatro niveles. A saber, aplicación, transporte, Internet e interfaz de red. Naturalmente, cada uno de estos niveles tiene su correspondencia con uno o varios niveles del modelo OSI (Figura 1).

La familia de protocolos TCP/IP tiene dos grandes virtudes: su escalabilidad y la capacidad de conectar todo con todo. Podemos poner en red millones de ordenadores, cada uno de su padre y de su madre, y conseguir que trabajen más o menos juntos (eso es justamente lo que ocurre con Internet). Por si esto fuera poco, esos ordenadores ni siquiera comparten el mismo modo de conectarse y las conexiones pueden ir desde un cable de serie hasta vía satélite. En fin, un desastre.

A *grosso modo,* TCP/IP pone cierto orden en este caos transmitiendo los datos empaquetados. Cada paquete comienza con una cabecera que contiene información de control y a continuación los datos.

El modelo OSI

Antes de ver más detalladamente los protocolos que componen TCP/IP, vamos a recordar someramente el modelo OSI (Interconexión de Sistemas Abiertos), ya que de esta manera comprenderemos mucho mejor el funcionamiento de TCP/IP y cómo actúa cada protocolo.

El modelo OSI se estructura en siete niveles establecidos por situaciones en las que se requiere un nivel distinto de abstracción. Cada nivel realiza funciones definidas para las que se normalizan protocolos. Estos niveles son el físico, enlace, red, transporte, sesión, presentación y aplicación.

- El nivel físico se ocupa de los aspectos físicos de la conexión, como son el medio de transmisión empleado, la temporización de las señales, el comienzo y el fin de la transmisión, etc.

- El nivel de enlace se ocupará de que el *hardware* esté bien configurado, de la reinicialización y de que sea compatible. Es decir, asegura una transmisión libre de errores. También establecerá mecanismos para regular el flujo de datos y agrupará el flujo de datos en unidades que llamamos tramas.

- El nivel de red encamina los paquetes de datos. Usa algoritmos para localizar el mejor encaminamiento hacia los nodos y funciona con direcciones lógicas suministradas por la tarjeta de red (están grabadas en la EPROM de la tarjeta). Estas direcciones son las llamadas *node address.*

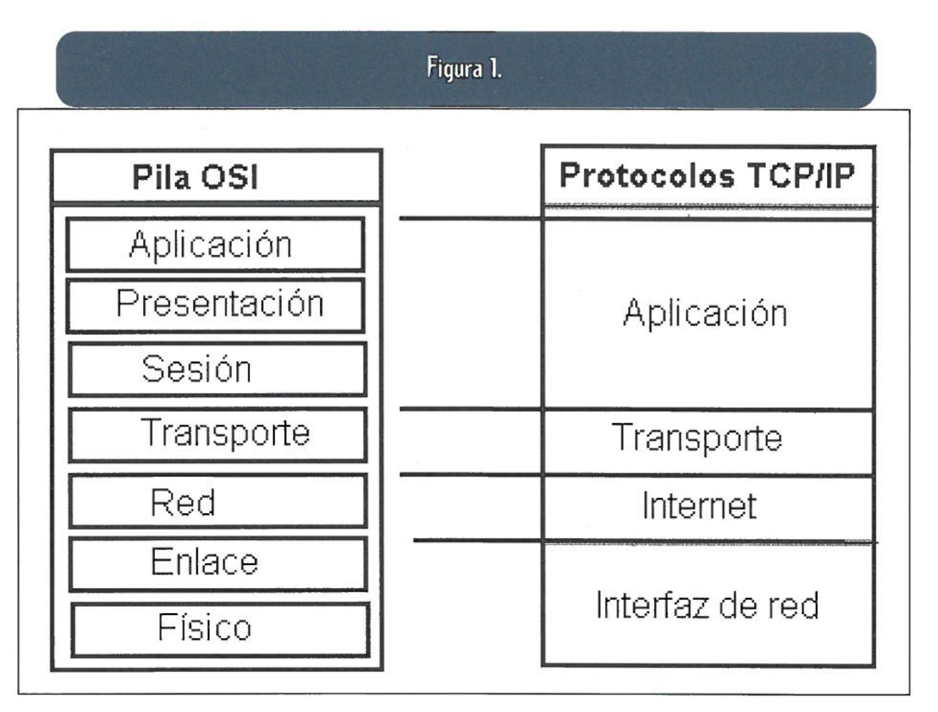
- Transporte. Este nivel proporciona a la capa superior un servicio de transporte de datos que ha de ser fiable. Se encarga de las señales de reconocimiento, de validación, de ordenar los datos que llegan y establecer el control de flujo, entre otras funciones.

- El nivel de sesión proporciona un servicio de sincronización que permite recuperar la sesión cuando se ha producido una caída de la conexión, sin que se dupliquen ni se pierdan datos. También gestiona los testigos de comunicación, de modo que los procesos sean capaces de comunicarse entre sí de forma ordenada. Asimismo, se ocupa de desensamblar y desencriptar los paquetes.

- El nivel de presentación permite que sistemas que no utilizan el mismo formato de representación de datos puedan comunicarse. También proporciona compresión de los datos para aumentar la eficiencia de la red. Se proporciona también el cifrado de los datos, si bien no es exclusivo de este nivel, ya que puede ser aplicado en otros.

- El nivel de aplicación proporciona servicios como el correo electrónico, servicios de directorio, llamadas a procedimientos remotos, etc.

Figura 1.



El protocolo IP

El *Internet Protocol* (IP) es un protocolo a nivel de red en la pila OSI que permite que las aplicaciones se ejecuten de forma transparente sobre las redes interconectadas y facilita la entrega de paquetes para los demás protocolos de la familia TCP/IP. De esta forma se soluciona el problema de las conexiones, ya que las aplicaciones no necesitan saber qué *hardware* está siendo utilizado en la red. Tanto da que la topología sea Ethernet o Token Ring. IP no garantiza que los paquetes lleguen a su destino, ni su orden. Tan sólo garantiza la integridad del encabezado IP, por lo que la fiabilidad de los datos debe estar asegurada en protocolos de nivel superior.

El protocolo TCP

Por su parte, el *Transmission Control Protocol* (TCP) se ocupará de que los datos sean efectivamente entregados, que lo que se recibe se corresponde con lo que se ha enviado y que los paquetes sean reensamblados en el orden en que fueron enviados. Si hubiera algún problema con la transmisión de los paquetes, TCP se encargará de volver a enviarlo. Esta fiabilidad convierten hacen de TCP un protocolo *pesado*, ya que esta fiabilidad se paga con un mayor tráfico en la red debido al envío de las señales de confirmación (ACK). Así pues, estamos sacrificando la velocidad en la transmisión de los datos en pro de la fiabilidad, por lo que este protocolo es ideal para aquellas aplicaciones en las que no podemos permitirnos perder un solo *byte*, como aplicaciones cliente-servidor de base de datos o el correo electrónico.

El protocolo UDP

UDP (Protocolo de datagramas de usuario) será el protocolo a utilizar si la fiabilidad no es un factor crítico o disponemos de un buen cableado que garantice una mínima pérdida de paquetes.

UDP no garantiza la entrega de los paquetes ni el orden correcto de los paquetes recibidos. Un buen ejemplo de la sabia utilización de UDP es jugar al *Quake*, en red. Si se pierde el envío de la pulsación de una tecla por parte de un jugador, pues se ha perdido. No pasa nada.

Juegos aparte, podemos solventar las carencias de UDP en niveles superio-

res de red, esto es, a nivel de aplicación, y establecer allí los controles para la comprobación de datos. Además, podemos establecer optativamente las sumas de comprobación de datos en el protocolo UDP, éstas validarán la integridad de los encabezados y los datos. Aún así, el protocolo será mucho más ligero que el TCP.

Tabla 1. Construcción de direcciones IP.

Clase	Valor Primer byte	Id de Red	Id de Host	Nº de Redes	Nº de Hosts
A	1-126	W	x.y.z	126	16.777.214
B	128-191	w.x	y.z	16.384	65.534
C	192-233	w.x.y	Z	2.097.151	254

El protocolo ARP

El protocolo ARP (Protocolo de resolución de direcciones) no se dedica específicamente al transporte de datos. Su misión consiste en conseguir la dirección física del destinatario de un paquete IP. Cuando un ordenador quiere comunicarse con otro bajo TCP/IP, el sistema que envía el paquete debe consignar la dirección del destinatario. El protocolo IP consigue esta dirección mediante la difusión de un paquete de petición ARP que contiene la dirección del sistema con el cual se quiere comunicar. Todos los ordenadores de la red detectan estas difusiones, y aquel que tenga la dirección IP solicitada transmitirá su dirección al sistema que la solicitó mediante un paquete de respuesta ARP. Una vez establecida la relación entre la dirección física y la dirección IP, el sistema solicitante la almacenará en una tabla para su uso posterior en otras comunicaciones. El paquete de respuesta ARP puede ser leído por otros ordenadores, que aprovecharán para actualizar sus tablas ARP. De este modo se evita cargar la red enviando difusiones una y otra vez.

El protocolo ICMP

ICMP (Protocolo de mensajes de control de Internet) tampoco está directamente relacionado con el envío de datos. ICMP está encapsulado en los paquetes IP y permite que los sistemas de una red IP compartan información relativa a errores y estado, por lo que es útil para resolver problemas de transmisión.

La utilidad *ping*, por ejemplo, utiliza los paquetes ICMP para determinar si un sistema está en uso.

Direcciones IP

Las direcciones IP permiten que el envío de datos entre ordenadores se haga de forma eficaz, de forma parecida al uso de los números de teléfono en las llamadas telefónicas.

Para que todo funcione correctamente deberemos suministrar tres valores: dirección IP, máscara de subred y *gateway* predeterminado.

La dirección IP tiene 32 bits, repartidos en cuatro campos de 8 bits separados por puntos. Cada uno de estos campos puede tener un valor comprendido entre 0 y 255. Cada ordenador o dispositivo conectado a una red IP deberá tener una dirección única en la red, por ejemplo 132.100.100.1.

La dirección IP aporta en realidad dos datos el identificador de la red y el identificador del *host*. El identificador de la red agrupa los sistemas que pertenecen a una misma red física. Por tanto, todos los sistemas de una red deben tener el mismo identificador de red, y este ha de ser único en el conjunto de redes. El identificador del *host* distingue cada sistema o dispositivo dentro de la red y ha de ser único dentro de la red.

Para poder ajustarse a redes de distintos tamaños, la comunidad de Internet estableció tres clases de redes, que son: A, B y C. La diferencia está en el número de ordenadores que podrá tener la red, y se distinguen por el número de bytes que ocupará el identificador de red. Así, la máscara de subred de la clase A será 255.0.0.0, y podrá tener 16.777.214 ordenadores. Una red de clase B tendrá la máscara 255.255.0.0 y puede tener 65.534 ordenadores. Finalmente, una red de clase C tendrá la máscara 255.255.255.0 y puede tener 254 ordenadores. El número de *host* indica el número del ordenador dentro de la red. Por ejemplo al primer ordenador de la red 200.90.80 le asignaremos la dirección 200.90.80.1.

Distinguiremos las tres clases de redes por el valor que le damos al primer byte de la dirección. En la tabla 1 podéis ver la estructura de las direcciones IP y cómo se forman. Las letras *w x y z* designan los cuatro bytes que forman la dirección IP.

Para que todo nos funcione correctamente, deberemos dar por fin un valor de *gateway* predeterminado. La dirección del *gateway* o encaminador identifica al ordenador que está conectado a la subred local y a otras redes. El encaminador conoce los identificadores de red de las otras redes y cómo llegar hasta ellas.

El lector avispado se habrá dado cuenta de que no están todas las que son. Efectivamente, hay algunas direcciones que están reservadas. La dirección 127 de red (el primer byte de la dirección) está reservada para las pruebas de bucle cerrado y las comunicaciones entre procesos dentro de un equipo local. Las direcciones 234 y siguientes están reservadas para protocolos especiales, el valor 255 en el identificador de *host* está reser-

vado para *broadcast*, mensajes que se envían a todos los sistemas de la red. Finalmente, el valor 0 está reservado para ordenadores que aún no conocen su dirección, ya sea la red o subred a la que pertenecen, su propio número de *host* o ambas.

Estas tres clases de redes son más flexibles de lo que parecen a simple vista. Si tuviéramos una red muy grande podríamos establecer una red de tipo B y dividirla en *subnets*, simulando internamente redes de tipo C.

Configuración de TCP/IP

Con lo visto hasta ahora estamos en condiciones de configurar los principales parámetros para el protocolo TCP/IP. En primer lugar abriremos el panel de control y efectuaremos una doble pulsación sobre el icono *Red*. Pinchamos en la pestaña *Protocolos* y seleccionamos TCP/IP. En caso de que no aparezca lo añadiremos mediante el botón *Agregar*. Una vez seleccionado el protocolo efectuaremos una doble pulsación sobre él o pincharemos el botón *Propiedades*. Nos aparecerá una ventana como en la figura 2. En nuestra futura intranet este será el servidor para los servicios de Web, FTP y Gopher, así pues lo prepararemos como tal. En las propiedades del protocolo configuraremos tantas direcciones IP como tarjetas tenga el servidor. En este caso sólo tenemos una y un módem. Como este será el servidor, haremos que sea el primero de la red y le asignamos una dirección IP de la clase C, que podría ser perfectamente válida para salir a Internet en un futuro. Aunque esto es harina de otro costal, ya que tendríamos que solicitar a Internic una dirección IP única para Internet; pero de momento ya nos vale tener una red de clase C. En este caso le hemos dado 195.180.123.1. Si fuéramos clientes de otro servidor tendríamos que especificar la dirección IP que nos hubiera asignado el administrador o bien marcar la opción *Obtener dirección IP de un servidor DHCP* para que el servidor DHCP nos asignase una automáticamente.

Lo siguiente que vamos a especificar es la máscara de subred, como hemos dicho, es de clase C; así que ponemos 255.255.255.0.

En *Dirección de gateway o puerta de enlace predeterminada* pondremos la dirección de nuevo 195.180.123.1, ya que es este el ordenador que tiene acceso a otras redes. De no ser así pondríamos la dirección IP del ordenador que tiene acceso a otras redes. Si no lo especificáramos no podríamos sacar los paquetes IP de nuestra propia red, a no ser que especificásemos una ruta mediante el comando *route* de TCP/IP.

Si pulsamos sobre el botón *Avanzadas* podremos especificar hasta cinco direcciones IP adicionales por cada adaptador y otras tantas para el *gateway*. Esto puede sernos útil para crear varias redes IP lógicas dentro de una sola red física.

De momento, los parámetros ya configurados del protocolo TCP/IP nos valen para ir tirando. Según vayamos añadiendo funcionalidades iremos completando la configuración del protocolo.

Una vez resuelta la dirección IP del servidor, vamos a meternos con las direcciones de los clientes.

Figura 2.

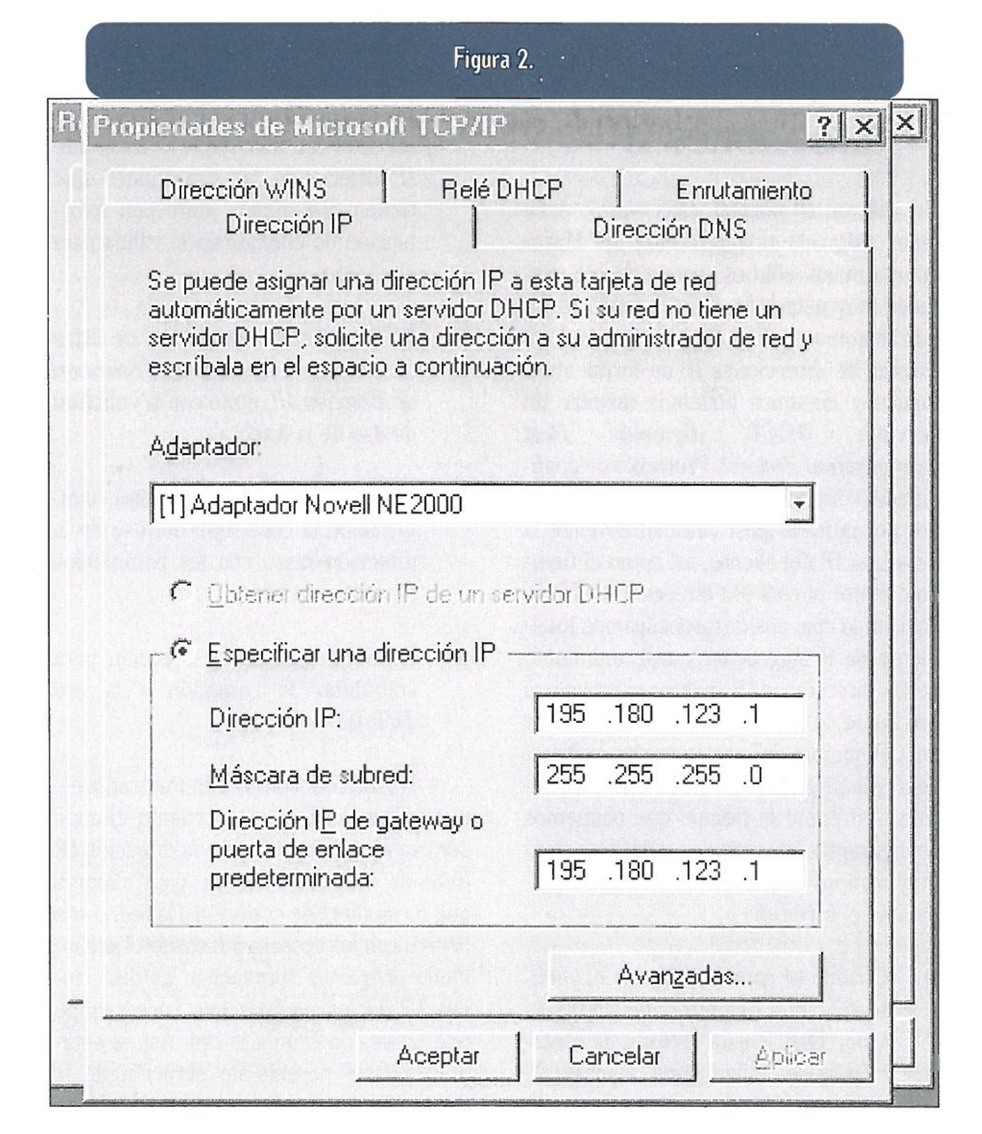
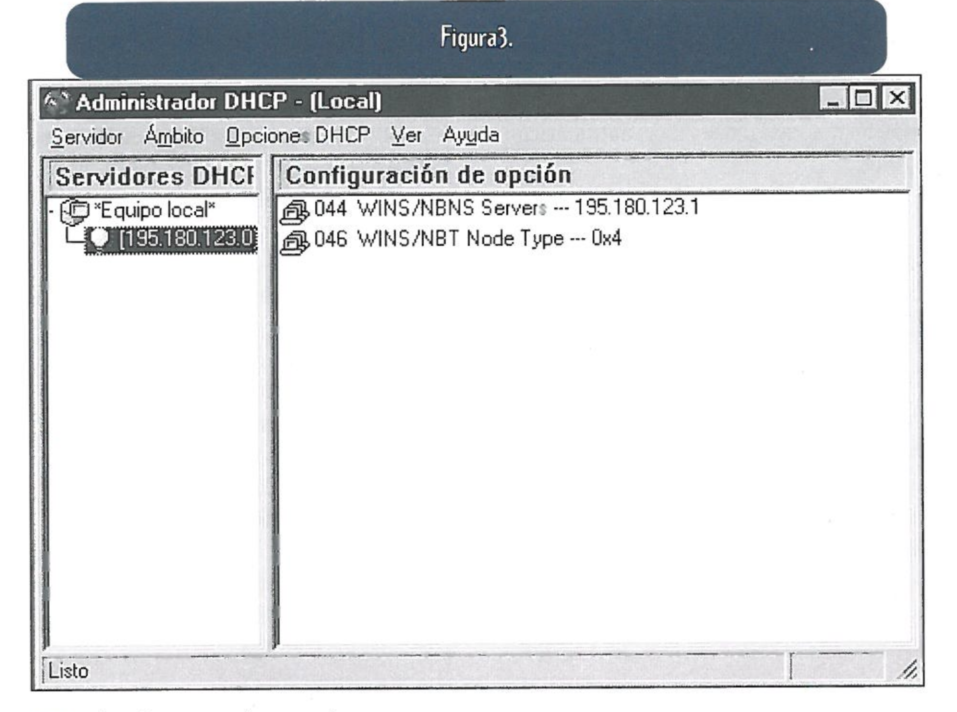


Figura3.



Asignación de direcciones IP

En una red IP normal, cada equipo debe tener asignada una dirección IP. Hacer esto 'a mano' sólo es recomendable para redes muy pequeñas, de otro modo resultará engorroso e inducirá a errores. Para asignar las direcciones IP de forma automática y dinámica podemos instalar un servidor DHCP (*Dynamic Host Configuration Protocol*, Protocolo de configuración dinámica de *host*). Este protocolo nos permitirá asignar automáticamente la dirección IP del cliente, así como el tiempo durante el cual esa dirección será válida. Con lo cual nos despreocupamos totalmente de la asignación y mantenimiento de las direcciones IP. Incluso si movemos un cliente de una subred a otra, el servidor DHCP liberará automáticamente la dirección y le asignará otra en su nueva ubicación. Así pues, el tiempo que ocupemos configurando el servidor DHCP, estará muy bien invertido. Veamos cómo funciona todo el invento:

- Cuando se conecte a la red, el cliente solicitará una dirección IP al servidor DHCP, esta *Petición de concesión IP* se enviará como mensaje de difusión en la red.
- Todos los servidores DHCP de la red que reciban este mensaje contestarán con una *Oferta de concesión IP* ofreciendo las direcciones que tienen disponibles junto con información de configuración válida para el cliente.
- El cliente seleccionará una de éstas direcciones y enviará una *Selección de dirección IP*, junto con la solicitud de uso de la dirección.
- El servidor DHCP contestará confirmando la concesión del uso de la dirección junto con los parámetros de configuración.
- El cliente utiliza la dirección para completar la conexión a la red TCP/IP.

Nosotros, como administradores, especificaremos durante cuánto tiempo será valida la concesión de la dirección IP. Esto es útil, por ejemplo, para clientes que no siempre se conectan a la red, como ocurre con los accesos a Internet. De este modo podremos aprovechar las direcciones IP de los equipos que hace tiempo que no se conectan a la red. Así, si establecemos el periodo de duración de la concesión en tres días, las direcciones de clientes que hayan superado ese periodo de tiempo sin renovar la solicitud quedarán liberadas. La negociación de las renovaciones se produce de la siguiente manera:

- Cuando ha transcurrido el 50% del tiempo asignado a la concesión, el cliente solicitará su renovación al mismo servidor DHCP que se la concedió.
- Si por cualquier motivo no es posible comunicar con dicho servidor y ya ha transcurrido el 87% del tiempo de concesión, el cliente intentará renovarla enviando un mensaje de petición a todos los servidores DHCP disponibles.
- Si la concesión caduca y el cliente no ha logrado una nueva concesión, dejará de usar inmediatamente la dirección que tenía asignada y enviará una nueva *Petición de concesión IP*.

Ahora vamos a ver como hacer que funcione el DHCP, esa maravilla de la técnica.

Instalación de DHCP

Para instalar el servidor de DHCP ejecutaremos el icono *Red* del *Panel de control*, seleccionaremos la ficha *Servicios* y pulsaremos el botón *Agregar*; seleccionaremos *Servidor de DHCP*. Tras la copia de archivos nos preguntará por la dirección IP del servidor DHCP. Una vez se la demos, debemos reiniciar el equipo, tras lo cual veremos que se ha añadido el nuevo servicio y que en las *Herramientas administrativas* disponemos de una nueva opción llamada *Administrador de DHCP*.

Antes de que los clientes puedan obtener una dirección IP deberemos crear un ámbito. El ámbito es un grupo de equipos que ejecutan el servicio cliente de DHCP en una subred, y se utiliza para

definir los parámetros para cada subred. Los ámbitos utilizan una única máscara de subred para determinar la subred asociada con una dirección IP determinada.

Para crearlo, ejecutaremos el *Administrador de DHCP*. En esta aplicación se nos muestra el servidor DHCP, que en este caso se nos indica como máquina local (figura 3). Pulsaremos dos veces sobre esta entrada y abriremos el menú *Ámbito*, donde seleccionamos la opción *Crear*. Veremos una pantalla como en la figura 4. En ella especificaremos los siguientes valores:

- *Inicio*. Especificaremos la primera dirección IP que se puede conceder. En este caso 195.180.123.2

- *Fin*. La última dirección IP que se puede conceder. Pondremos 195.180.123.200

- *Máscara de subred*. Indicaremos la máscara de subred de nuestra red. En nuestro caso es de tipo C, esto es, 255.255.255.0.

En el bloque intervalo de exclusión indicaremos las direcciones serán adjudicadas a nadie por pertenecer a otros servidores DHCP, clientes que no utilizarán este servicio, estaciones de trabajo sin discos o clientes de acceso remoto. En nuestro caso, no pensamos tener esas cosas tan raras, pero nos reservamos algunas por si acaso. Así pues, en *Inicio* especificamos 195.180.123.2 y en *Fin* 195.180.123.10. Luego pulsamos el botón *Agregar* para añadirlas a la lista de direcciones excluidas. Podemos repetir el proceso para establecer otros intervalos. Si deseamos irlas excluyendo de una en una ponemos la dirección en *Inicio* y dejamos *Fin* en blanco. Para eliminarlas de la lista de direcciones excluidas, las seleccionamos en la lista y pulsamos el botón eliminar. En el bloque *Duración de la concesión* especificaremos:

- *Sin límite*. La concesión de direcciones no caducará nunca.

- *Limitado*. Para definir el límite de tiempo de la concesión en días, horas y minutos.

Asimismo, podemos especificar un nombre y un comentario para el ámbito. Una vez finalicemos, confirmaremos que deseamos activar el ámbito, que aparecerá en el lado izquierdo de la ventana.

Las concesiones activas

Para ver las concesiones que tenemos activas, abriremos el menú *Ámbito* y seleccionamos *Concesiones activas*. Veremos una pantalla como en la figura 5, donde se muestran las direcciones IP concedidas. Si marcamos propiedades podremos ver el identificador único, que especifica la dirección de control de acceso de la tarjeta adaptadora de red del equipo cliente, el nombre del cliente y el tiempo que durará la concesión.

Si la concesión es reservada podremos realizar modificaciones en todos los apartados, a excepción de la dirección IP.

También podremos verificar el estado de la base de datos DHCP, mostrar sólo las direcciones de los clientes que utilizan direcciones reservadas, ordenarlas por dirección IP o por nombre y eliminar concesiones.

Si deseamos que algún cliente tenga una dirección IP fija lo que podemos hacer es reservar varias direcciones IP. Para hacer esto, pulsaremos la opción apropiada, utilizando la opción *Agregar reservas* del menú *Ámbito* e indicaremos los siguientes parámetros:

- Dirección IP que deseamos reservar. No debe estar dentro de las que ya habíamos excluido inicialmente para el ámbito, y si indicamos una dirección ya concedida se eliminará la concesión del cliente anterior.

Figura 4.

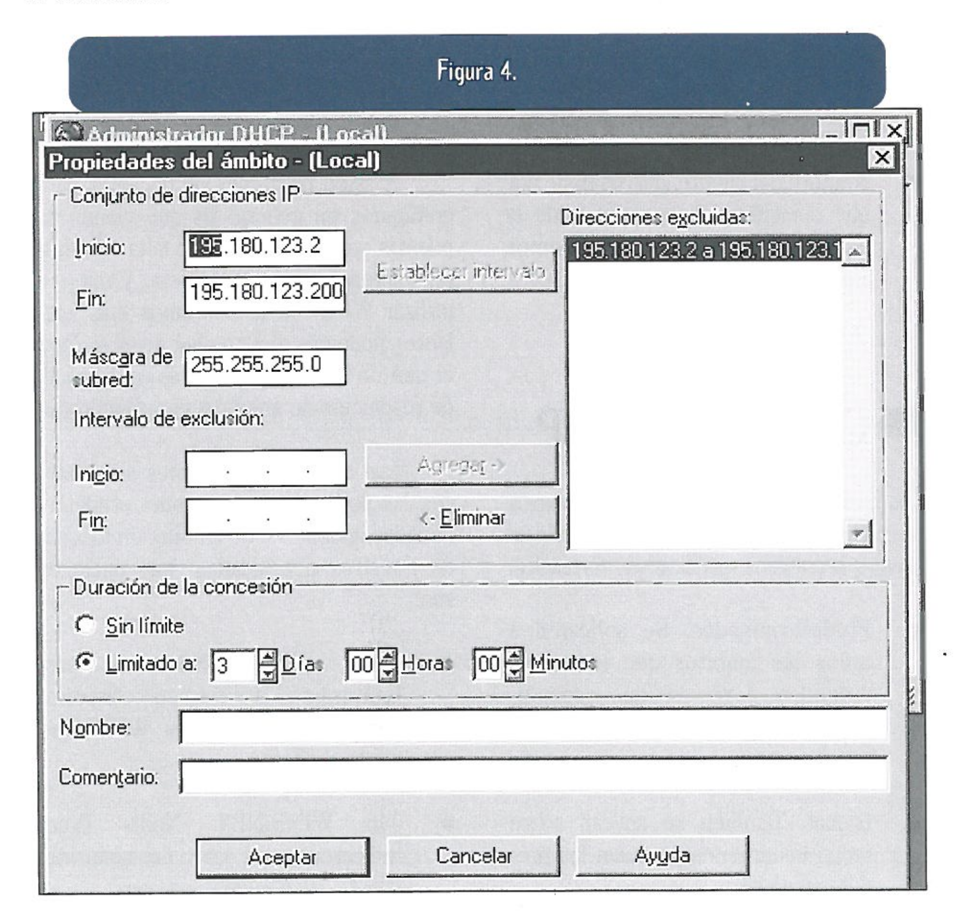
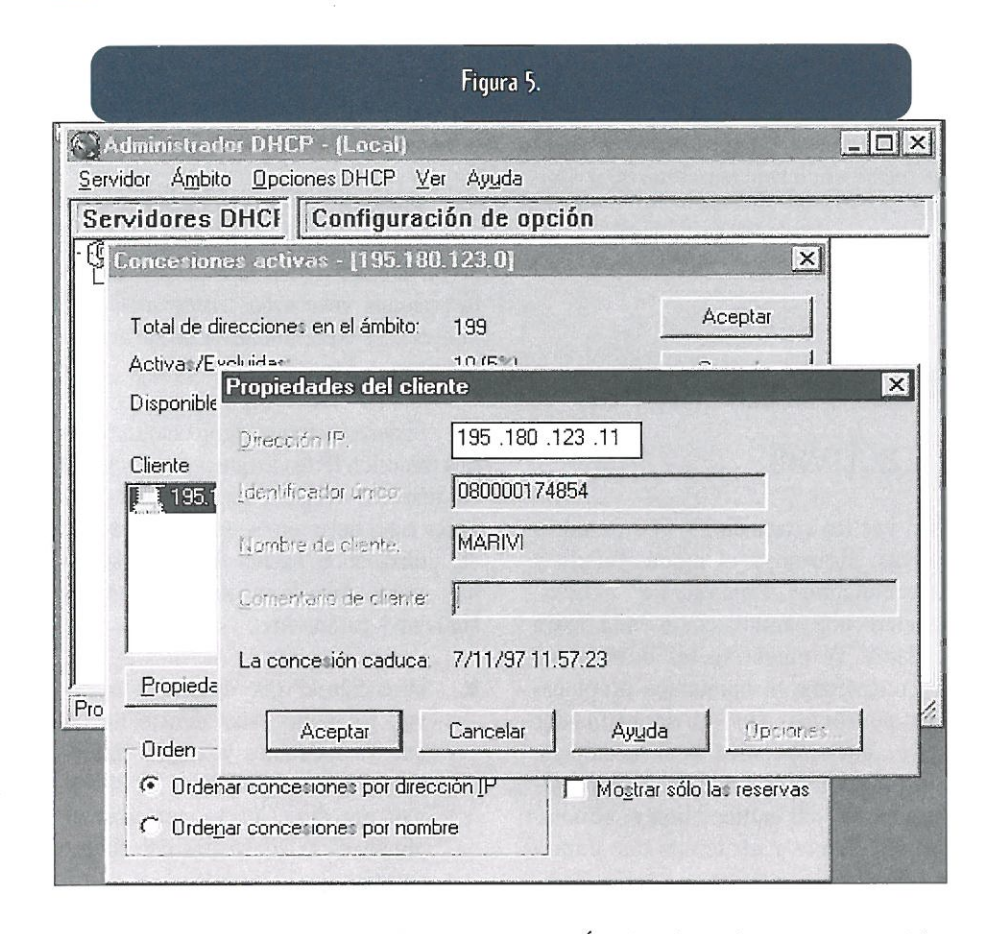


Figura 5.



- Identificador único la tarjeta de red del cliente (lo podemos averiguar tecleando NET CONFIG WKSTA).

- Nombre del cliente, que no tiene por qué coincidir con el nombre de la estación. Opcionalmente podemos agregar un comentario.

Opciones DHCP

Las opciones de DHCP son los parámetros de configuración que el servidor asignará a un cliente. Se pueden especificar en tres niveles:

- Predeterminados. Se aplicarán a todos los ámbitos que tengamos definidos, a no ser que especifiquemos opciones globales o de ámbito.

- Global. También se aplican sobre todos los ámbitos y anulan las predeterminadas.

- Ámbito. Se aplican para un ámbito específico y anula la global y la predeterminada.

Existen montones de opciones para configurar, en general las que vienen ya puestas van bien y no hace falta tocarlas, pero si tenemos algún problema y vamos a utilizar WINS (lo instalaremos más adelante), podemos tocar un par de ellas. Para el que no le suene, WINS es un servicio de resolución de nombres para Windows.

Las opciones que vamos a establecer, las podemos indicar como predeterminadas, globales o de ámbito, en función de nuestras necesidades. Las opciones son:

- 044 WINS/NBNS Servers. Indicaremos la dirección o direcciones de los servidores WINS que usará el cliente.

- 046 WINS/NBT Node Type. Indicaremos el tipo de nodo del cliente, este nodo puede ser:

  - 0x1 B-nodo. El cliente utiliza mensajes, pero limitados a la red local (para versiones antiguas de Microsoft).

  - 0x2 P-nodo. El cliente ni crea ni responde mensajes. Los equipos registrados en un servidor WINS utilizan comunicaciones punto a punto. Si el servidor WINS no está disponible, no podrán registrarse en la red.

  - 0x4 M-nodo. El cliente intentará conectarse como B-nodo, si lo consigue intentará cambiarse a P-nodo, ya que B-nodo genera un tráfico de mensajes elevado en la red.

  - 0x8 H-nodo. Es el más moderno. Lo soportan Windows NT y Windows 3.11. El cliente intentará conectarse como P-nodo, y si no lo consigue intentará registrarse como B-nodo.



¿Por qué TCP/IP?

Si tenemos una red local más o menos pequeña y monolítica, en cuanto al sistema operativo se refiere, lo mejor es utilizar el protocolo que nos ponga por defecto el fabricante del sistema operativo de red. Es decir, si utilizamos Windows NT y los clientes son sistemas operativos Microsoft, lo mejor será utilizar NetBEUI; un protocolo muy pequeño y muy eficiente. Si utilizamos Novell Netware, lo suyo será poner IPX/SPX. En el momento que empecemos a mezclar diferentes sistemas operativos, lo mejor será implementar TCP/IP. Pese a que es un protocolo *pesado*, siempre es mejor que tener dos o tres protocolos montados para conectar los diferentes segmentos de red. TCP/IP es un protocolo compatible con todos los sistemas operativos y nos permite comunicar y transferir datos entre sistemas muy diferentes. Además, nos permitirá la salida a Internet o bien montar una intranet.

Lo de la intranet lo veremos más adelante con más detalle, y lo de Internet... Pues empieza a ser necesario por cuestiones de imagen, al margen de los innegables beneficios que proporciona.

Un servidor recuerda que en cierta ocasión tuvo que hacer una factura para un cliente con la máquina de escribir, ya que se había estropeado el único ordenador que teníamos. Como las máquinas de escribir no tienen centrado de párrafo, ni corrector (bueno, el Typpex), ni nada, la cosa se demoró un poco y el cliente se marchó bastante indignado exclamando *¡Qué se puede esperar de una empresa donde todavía utilizan máquinas de escribir!* Pues bien, ya hemos subido un escalón y... ¿Qué se puede esperar de una empresa que no tiene e-mail, o página Web? Queramos o no, de aquí a muy poco el que no sea accesible a través de Internet parecerá tonto, lo cual no quiere decir que lo sea. Pero es posible que nos vean 'con máquinas de escribir'.

Las intranets

La salida de nuestra empresa a Internet es conveniente, bien por necesidad o por cuestiones de imagen, pero tener una intranet es algo absolutamente bueno. Se mire como se mire.

Las intranets son básicamente modelos a escala de Internet, están basadas en la misma tecnología y protocolos que han hecho posible el éxito de Internet. Una intranet nos proporcionará mayor agilidad en la empresa, mayor calidad y la oportunidad de manejar más información más rápido, todo ello aliñado con una no desdeñable reducción de costes.

Mediante una intranet podremos organizar una reunión en segundos; tan sólo hemos de enviar un mensaje por correo electrónico a los participantes. Los *memos* distribuidos por toda la oficina en papeles pegados a la pared, se acabaron, sólo hay que poner el documento en nuestro servidor de Web. De igual modo, nuestros comerciales podrán ofrecer la ultimísima oferta actualizada utilizando su navegador, ya que para dar a conocer un documento a toda la empresa sólo hay que crearlo y migrarlo a nuestra red interna. Si abrimos nuestra red, esta comunicación rápida y eficaz puede trasladarse a nuestros clientes, proveedores, etc.

Debemos cuidar mucho la seguridad si deseamos abrir nuestra empresa a Internet.

Una intranet bien diseñada nos ahorrará tiempo y dinero, y seguramente cambiará la percepción que los usuarios tienen acerca de la informática. La intranet descansará sobre uno o varios servidores, y facilitará el acceso a los datos de una forma muy intuitiva, de modo que la formación de los usuarios para sacar el máximo provecho no será un problema.

Los beneficios que nos ofrece una intranet se pueden resumir en los siguientes puntos:

- Rápida instalación, que se puede realizar en unos pocos días o incluso algunas horas.
- Mínima inversión.
- Es más eficiente y más barata que los métodos tradicionales de distribución de información basados en papel.
- Crearemos una arquitectura de plataforma abierta, con lo cual podremos añadir cualquier aplicación y crearlas con independencia del cliente (java).
- Es escalable, y, por tanto, crecerá sin problemas tanto como lo hagamos nosotros.
- Podremos extenderla con aplicaciones multimedia con sonido, videoconferencia, etc.
- El usuario necesitará una formación mínima para hacerse con el entorno.

Desgraciadamente las intranets tienen su punto flaco en la seguridad. Lo normal es que finalmente demos salida a Internet, exponiendo de esta manera nuestra red a riesgos de seguridad. La cosa está en que acceder a los datos que el usuario necesita sea muy fácil, pero que no pueda tener acceso a información que no hayamos autorizado expresamente. Para conseguirlo debemos poner especial cuidado en la seguridad de las bases de datos y de los servidores. En cuanto a las bases de datos deberemos establecer un férreo control de los accesos y combinarlo con los privilegios y permisos propios del sistema operativo de red. En cuanto al servidor, lo mejor es situar los datos más delicados en un servidor que esté tras un cortafuegos. Si también tenemos dinero en juego habría que implementar técnicas criptográficas en la comunicación entre los navegadores y los servidores, utilizando sistemas como el SSL (*Secure Sockets Layer*) o SET (*Secure Electronic Transaction*).

Como hemos comentado antes, el desarrollo natural de la intranet es la extranet. Básicamente, una extranet consiste en abrir nuestra intranet a otras empresas, clientes, distribuidores, etc. También para comunicar las diferentes delegaciones de nuestra empresa, creando de esta manera un único espacio donde todos tienen acceso a la información al mismo tiempo. Estén donde estén.

Próximo número

El mes que viene humanizaremos las direcciones IP y nos centraremos en la resolución de nombres de Internet, así como en los problemas que nos planteará el enrutamiento.

Sistemas Distribuidos II: Arquitectura

Jorge F. Delgado Mendoza
ernar@convex.es

Los Sistemas Distribuidos son uno de los avances más interesantes de los últimos años en el campo de la Informática. Las redes de alta velocidad y la aparición de procesadores de bajo precio, hacen factible construir sistemas de gran capacidad de cálculo a partir de ordenadores más pequeños.

En los últimos años se está produciendo una auténtica revolución en el campo de la Informática. La aparición de nuevos tipos de aplicaciones, - Multimedia, Realidad Virtual, Reconocimiento de Voz, etc... - todos los cuales tienen en común una necesidad casi exagerada de recursos, han puesto sobre la mesa las limitaciones de las arquitecturas clásicas a la hora de abordar soluciones para este tipo de problemas.

En el artículo anterior, el primero de la serie, identificamos los puntos en los que los sistemas clásicos se encontraban más limitados, a pesar de rápida evolución que hoy en día está experimentado la tecnología de microprocesadores. Tras un detallado análisis llegamos a la conclusión de que las carencias más acusadas de los sistemas clásicos eran cuatro:

- Falta de Capacidad de Proceso.
- Baja Conectividad.
- Interfaz de Usuario Rudimentaria.
- Poca Heterogeneidad.

Vimos también como dos de ellos estaban resueltos *a priori* en un Sistema Distribuido el cual, por definición, consta de un número arbitrario de máquinas cuyas arquitecturas no tienen porqué ser similares. De esta manera se resolvían los problemas de Conectividad y Heterogeneidad. En cuanto a la Capacidad de Proceso, bastaba con hacer que todos los miembros del Sistema Distribuido trabajasen en paralelo para poder aumentarla arbitrariamente. Finalmente, vimos como el concepto de Interfaz de Usuario es ortogonal al de Sistema Distribuido lo cual, si bien no mejoraba nuestra situación, tampoco la empeoraba.

Acabado este análisis, profundizamos en los tipos de soluciones que, desde dos puntos de vista completamente diferentes, se habían dado a la construcción de nuevos tipos de aplicaciones. Como ejemplos de ambos enfoques ofrecimos descripciones de sistemas que utilizan el paradigma *Cliente/Servidor*, así como de sistemas que ofrecían soluciones dedicadas, comparándolos con las posibilidades que nos ofrecen los Sistemas Distribuidos.

A continuación, definimos y explicamos el concepto más importante de un Sistema Distribuido: la *Máquina Virtual Única*. Este concepto permite ocultar al usuario, ya sea éste una persona o una aplicación, toda la complejidad de la red de microprocesadores que existe debajo. Las ventajas de esta aproximación son evidentes:

- El desarrollador no tiene por qué conocer las características intrínsecas de un sistema multiprocesador, pudiendo escribir sus programas como si de una máquina monoprocesador se tratara.

- El usuario puede abstraerse de las complejidades de la red que existe

debajo. Todas las aplicaciones son accesibles a él, independientemente de dónde se ejecuten o dónde se encuentren físicamente.

Una vez establecida la problemática y definidos los conceptos básicos, entramos en una detallada descripción de los fundamentos teóricos de la Arquitectura de Sistemas Distribuidos.

Comenzamos por una descripción de las diferentes topologías de red existentes, lo que se denomina la Arquitectura *Hardware*. Estudiamos las redes punto-a-punto, en anillo, en bus, jerárquicas, etc...

Una vez conectadas las máquinas, es necesario que se comuniquen entre ellas. El modelo de comunicación es siempre el paso de mensajes, sin embargo, el modelo que ve el usuario del Sistema Distribuido puede ser muy diferente en función de la Arquitectura *Software* del sistema. Nosotros clasificamos y estudiamos las diferentes posibilidades: *Send/Receive*, RPC's, Memoria Distribuida, etc...

Tras estudiar los diferentes modelos, llegamos a la conclusión de que, independientemente del tipo de red subyacente, la arquitectura *software* ideal es el Modelo de Memoria Distribuida, aunque sea el más complejo de implementar. Esta complejidad reside, como ya vimos, en la necesidad de que todas las copias de los datos se mantengan consistentes entre sí.

Esta necesidad nos condujo a estudiar los diferentes modelos que existen a la hora de mantener lo que denominamos *coherencia*. Vimos cómo las estrategias clásicas conducían a sistemas enormemente ineficientes, así como posibles soluciones a esta falta de rendimiento mediante la relajación del concepto de coherencia.

Para concluir el artículo anterior, estudiamos dos conceptos más: granularidad y replicación. El primero está relacionado con el tamaño mínimo de bloque capaz de moverse entre máquinas, mientras que el segundo define la estrategia que seguimos para mover y realizar copias de estos bloques. Ambos están estrechamente ligados con el rendimiento del sistema.

Muy bien, ahora ya conocemos los conceptos que hay alrededor de un Sistema Distribuido, pero no estamos ni un milímetro más cerca de poder construir, mucho menos de utilizar, un entorno de estas características. Ha llegado el momento de dar el siguiente paso.

Objetivos

El siguiente paso no es, ni más ni menos, que diseñar la arquitectura del Sistema Distribuido que vamos a implementar. Para ello vamos a utilizar los conocimientos adquiridos en el primer artículo de la serie, seleccionando en cada uno de los aspectos analizados la solución que más se adecue a nuestros propósitos.

¿Cuál es nuestro objetivo entonces? Bien; nuestra intención es construir un Sistema Distribuido lo suficientemente sencillo como para ilustrar de una manera didáctica los conceptos y las problemáticas de la programación distribuida, pero lo suficientemente complejo como para que sirva de algo. Al fin y al cabo queremos poder construir aplicaciones que utilicen sus servicios. Para ello, nuestro sistema ha de tener las siguientes características:

- Alto índice de conectividad entre diferentes plataformas, con especial atención a la portabilidad entre diferentes arquitecturas.
- Capacidad de tener un elevado número de usuarios utilizando el sistema, sin que se aprecien disminuciones en el rendimiento de las máquinas ni en la capacidad de la red que las une.
- Un alto grado de heterogeneidad, permitiendo la interconexión de máquinas de diferentes arquitecturas con diferentes sistemas operativos.
- Un diseño modular que nos permita independizar cada una de las capas de las aplicaciones que se construyan sobre él. De esta manera facilitaremos la construcción de interfaces de usuario que puedan ir evolucionando sin necesidad de cambiar nuestro sistema.

Requisitos: ¿Qué queremos conseguir?

El sistema será responsable de establecer y mantener los enlaces de datos entre los distintos miembros del Sistema Distribuido, así como de mantener la coherencia entre las distintas copias de la memoria distribuida. También será responsabilidad del sistema decidir y llevar a cabo la estrategia de replicación más adecuada a cada caso.

Todas las operaciones de comunicaciones y sincronización del sistema tendrán que ser llevadas a cabo mediante un protocolo ligero. Esta *ligereza* se concibe desde dos puntos de vista:

- *Uso de Recursos:* dado que uno de los problemas que queremos solucionar con la construcción de un Sistema Distribuido es la falta de capacidad de proceso, es inconcebible diseñar un sistema que ocupe la mayor parte del tiempo de CPU de cada una de las máquinas. La relación *I/O vs. CPU* debe ser alta, de tal manera que podamos realizar numerosas operaciones de comunicación con una ocupación muy baja de la CPU de las máquinas.
- *Acceso al Medio:* del mismo modo, es imprescindible que los mensajes de control de nuestro sistema no satu-

ren la red, debiéndose reservar casi todo el caudal disponible para las aplicaciones que se construyan sobre nuestro sistema.

Las unidades de información que el sistema presente a los usuarios del sistema, - es decir a los que deben implementar aplicaciones sobre el Sistema Distribuido, - deberán ser lo suficientemente flexibles como para representar numerosos tipos de datos, sin que con ello perdamos facilidad en la manipulación de la información.

Finalmente, y probablemente el punto más importante de todos, nuestro sistema deberá ocultar a las aplicaciones que lo utilicen y a los desarrolladores que construyan sus aplicaciones a partir de él la existencia de la red de comunicaciones. Para ello deberemos ofrecer una *API* de *Máquina Virtual Única*. De este modo, toda la experiencia que se tenga en la creación de interfaces y aplicaciones monousuario se podrá trasladar directamente a nuestro entorno distribuido.

Para poder satisfacer todos estos requisitos vamos a tener que renunciar a algunos aspectos importantes, aunque no imprescindibles, que caracterizan a los Sistemas Distribuidos. El más relevante de ellos, al que no sólo renunciamos en aras de satisfacer los objetivos expuestos sino también para conseguir un sistema más útil desde el punto de vista didáctico es la migración dinámica de procesos.

En nuestro sistema no vamos a distribuir la carga dinámicamente, sino que simplemente vamos a distribuir la información común a todos los procesos. De este modo vamos a disminuir enormemente la complejidad, sobre todo de cara a poder incorporar arquitecturas heterogéneas a nuestro sistema.

¿Qué es, pues, lo que queremos implementar? La respuesta es sencilla: un sistema de *Memoria Compartida Distribuida* que se comporte de manera idéntica a la memoria convencional, sobre la que podamos construir aplicaciones de todo tipo utilizando un número arbitrario de ordenadores y con un número arbitrario de usuarios.

Arquitectura: ¿Qué vamos a hacer?

La primera de las decisiones que tenemos que tomar es la de elegir el lenguaje de programación que vamos a utilizar a lo largo del desarrollo. Atendiendo a criterios de portabilidad y heterogeneidad, lo más apropiado será utilizar lenguaje C, ya que es uno de los lenguajes de programación más extendidos y posee una amplia gama de librerías estándar de comunicaciones.

Una vez elegido el lenguaje en el que realizar la implementación, deberemos decidir qué metodologías y modelos vamos a utilizar para los demás aspectos relevantes del sistema:

- Comunicaciones.
- Modelo de Consistencia.
- Modelo de Memoria.
- Granulado.
- Estrategia de Replicación.
- Integración.

Comunicaciones

Para cumplir con las condiciones impuestas de portabilidad y heterogeneidad la solución ideal es la utilización de protocolos de comunicaciones disponibles universalmente. Sin duda, la familia IP es la más extendida de todas, en sus dos vertientes: TCP/IP orientado a conexión y UDP/IP mediante datagramas.

La portabilidad está asegurada ya que es posible encontrar implementaciones de IP en las arquitecturas más comunes: Solaris, SunOS, Linux, Irix, HP-UX, DEC Unix, WindowsNT/95, etc... En cuanto a la heterogeneidad, la propia existencia de *Internet* - que cuenta con el mayor parque de máquinas diferentes en sus subredes - nos garantiza la posibilidad de que sistemas totalmente distintos se comuniquen entre sí.

Dentro de los protocolos IP, vamos a utilizar UDP en lugar del más conocido TCP. Esta decisión está condicionada por factores de rendimiento. La sobrecarga debida al establecimiento del canal virtual y el control de flujo asociados con comunicaciones TCP no son compatibles con una de las premisas básicas del sistema: ligereza en los flujos a través de la red.

Para que el sistema sea verdaderamente ilustrativo y mantenga su espíritu didáctico, vamos a utilizar el paradigma de *RPC (Remote Procedure Call)* en la implementación, en lugar de manipular directamente los *sockets*. De esta manera el diseño es más cómodo y nos ahorramos todo el trabajo relativo a control de flujo y exclusión mutua de accesos a la memoria compartida.

Modelo de Consistencia

Al describir los modelos de consistencia hacíamos una clasificación cuyo criterio venía dado por el tipo de concurrencia al que estaban destinados. Así teníamos dos tipos de modelos:

- *Modelos de Consistencia Sin Sincronización*: son aquellos orientados a mantener la coherencia de una memoria distribuida, independientemente del tipo de aplicación que los utilice.
- *Modelos de Consistencia Con Sincronización:* surgieron en como un intento de mejorar la eficiencia de los modelos anteriores, a cambio de limitar su utilización a regiones de exclusión mutua.

Los modelos *Con Sincronización* son más complejos en su funcionamiento y además exigen conocimientos *a-priori* por parte del programador que vaya a realizar aplicaciones sobre el sistema. Estos motivos hacen que, a pesar de ser más eficientes, no podamos emplearlos ya que van contra el requisito imprescindible de *Máquina Virtual Única*.

La diferencia entre los modelos entre los modelos que carecen de operaciones de sincronización radicaba en el equilibrio entre velocidad del sistema y conocimiento previo del tipo de información a manejar, así como los patrones de la misma.

En un extremo, la Consistencia Estricta exigía del sistema un comportamiento idéntico al que podíamos esperar de una máquina multiprocesador. En el otro, las consistencias PRAM y de Procesador relajaban al máximo el modelo, realizando un gran número de suposiciones que, en caso de no cumplirse, podía dar como resultado zonas de pérdida de coherencia.

Con el fin de acercar lo más posible el funcionamiento de la distribución de datos al de una máquina multiprocesador, vamos a utilizar una variante de la **Consistencia Secuencial**. Estudios empíricos sobre diferentes sistemas que utilicen este tipo de modelo de consistencia, demuestran que existe una cota superior de eficiencia en este modelo, expresable mediante la fórmula:

$r + w > t$

Es decir, si w es el tiempo de escrituras, r el de lecturas y t el tiempo mínimo entre nodos, aumentos en la velocidad de escrituras llevan consigo una disminución equivalente en la de lecturas y viceversa. Sin embargo, la experiencia demuestra que en la mayoría de las aplicaciones que vamos a querer construir sobre nuestro sistema, el número de lecturas supera en al menor un orden de magnitud al de escrituras. Nosotros vamos a aprovechar esta propiedad para disminuir arbitrariamente el tiempo de lecturas, con la esperanza de que una pequeña cantidad de escrituras no tenga un impacto excesivo en el rendimiento global del sistema.

Esta suposición, - porcentaje de lecturas muy elevado - es el precio a pagar para poder cumplir los requisitos de simplicidad y fácil integración contemplado en los requisitos del sistema.

Modelo de Memoria

Una vez seleccionado el paradigma de Memoria Distribuida como modelo a utilizar para la distribución de información, hemos de seleccionar cuál de entre todas las opciones posibles:

La primera de las opciones está basado en el concepto de *Página Distribuida*, el cual queda muy lejos del alcance de este artículo. El primero de los motivos es la enorme dificultad de implementar un sistema de estas características, ya que es necesario soporte en el propio *kernel (el núcleo)* del Sistema Operativo.

Además, el trasiego de páginas de una máquina a otra, aunque adecuado para algunos tipos de aplicaciones, es incapaz de cumplir los requisitos del sistema que pretendemos diseñar, ya que para conseguir una eficiencia aceptable sería necesario implementar un algoritmo de replicación de páginas muy complejo.

El modelo de *Objetos Distribuidos* es mucho más interesante. Está basado en objetos completos, con sus puntos de acceso y datos privados, los cuales migran a través del sistema. Sin embargo, la elección del lenguaje de programación C hace muy complicado el manejo de entidades de este nivel sin artificios innecesariamente complejos.

Nuestra elección va a ser un sistema intermedio entre ambos, que responde al concepto de *Variables Compartidas*. Su estructura no es de tan alto nivel como el sistema de objetos, pero permite una serie de generalizaciones que no son posibles en un modelo de páginas.

Finalmente, una vez decidido el sistema a utilizar, hemos de decidir si la distribución de datos va a ser total o parcial, ya que de ello van a depender el diseño e implementación del mismo.

- *Replicación Total:* todas las variables del sistema están presentes en todas las demás máquinas. Es el método más general e intuitivo, permitiéndonos implementar una *Máquina Virtual Única* de manera sencilla.

- *Replicación Parcial:* solo un grupo de variables está distribuido, mientras que todas las demás son locales en cada máquina.

En el sistema que vamos a construir en los artículos siguientes vamos a utilizar el segundo tipo; replicación parcial de variables. Los motivos son numerosos pero entre ellos destacan:

- Implementación más sencilla.

- Aunque el desarrollador de aplicaciones ha de saber que variables se comparte y cuáles no, la arquitectura interna de la red sigue estando oculta a él.

- Es más eficiente desde el punto de vista del rendimiento.

- Permite evitar errores de duplicación de parámetros en las diferentes máquinas.

Estrategia de Replicación

Para elegir una estrategia de replicación adecuada, hay que analizar el tipo de comportamiento con el que nos vamos a encontrar. Es decir, el modo y manera según el cuál las aplicaciones van a hacer uso de nuestro espacio de variables compartidas.

Un análisis detallado del tipo de aplicaciones que describimos en el primer

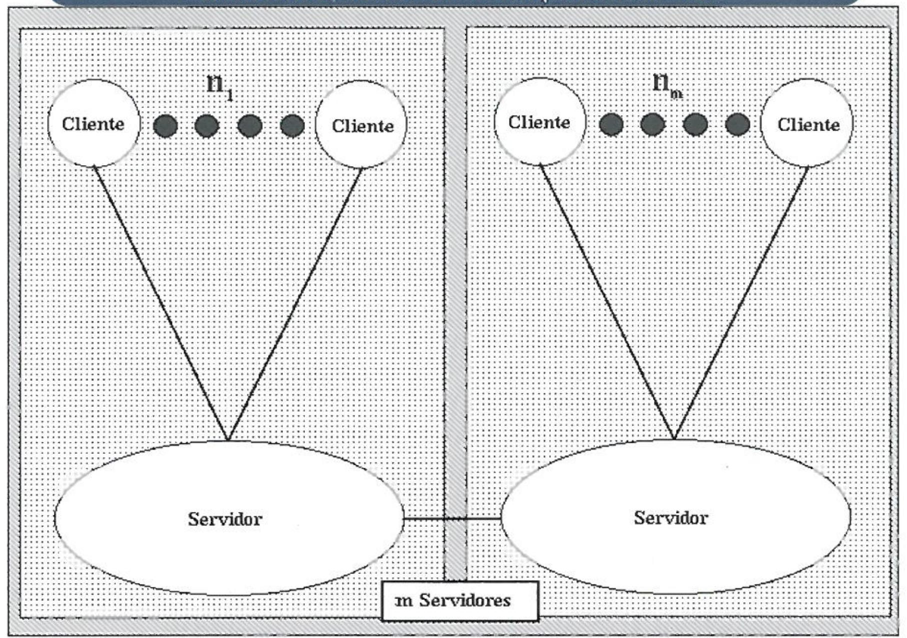


Figura 1.- Modelo estructural del Sistema Distribuido. Todos los clientes de una misma máquina se comunican con el mismo servidor. Existen m servidores diferentes comunicándose mediante RPC's no bloqueantes sobre UDP.

artículo de la serie nos permite llegar a las siguientes conclusiones:

- El *predominio de las lecturas* sobre las escrituras es evidente. Hay casos en los que la proporción llega a ser de tres órdenes de magnitud.

- Existe una gran *localidad de referencia* en los accesos a memoria por parte de las aplicaciones, es decir, los datos que se van a acceder en un futuro bien acaban de ser accedidos o bien están muy cerca de ellos en el mapa de memoria.

La segunda característica nos sugiere una gran caché de datos en cada uno de los nodos, ya que de esta manera minimizamos la mayor de las latencias, debida a los tiempos de circulación a través de la red.

La primera sugiere que debemos guardar la mayor cantidad de información posible en cada nodo, ya que el índice de lecturas es muy alto y es muy caro traerse la información de otro nodo.

Estos motivos, unidos a que el espacio de memoria distribuida es muy pequeño, nos permite adoptar la *replicación total* como estrategia de distribución de información. Es decir, vamos a mantener una copia actualizada del espacio de memoria distribuida por cada máquina que añadamos a la red.

Granularidad

Como ya vimos en el artículo anterior, escoger una estrategia de granulado adecuada es muy importante a la hora de diseñar un Sistema Distribuido serio. Por un lado, si los bloques de datos que migran son demasiado grandes, es posible llegar a la situación de vapuleo distribuido, en la que dos máquinas accedan constantemente a variables diferentes pero que están en la misma página. Por el otro lado, con un tamaño de bloque demasiado pequeño corremos el riesgo de que se produzca unan sobrecarga en la red que haga inviable el sistema.

En nuestro sistema, vamos a utilizar un grano muy pequeño, de una sola variable. Esta elección viene dada por criterios didácticos, ya que al tener una cantidad de variables muy pequeña, el número de mensajes de actualización no será nunca demasiado elevado.

Integración

Es imprescindible que las aplicaciones se comuniquen con el sistema de una manera natural, evitando que el programador deba realizar esfuerzos innecesarios a la hora de desarrollar sobre el Sistema Distribuido.

Con esta intención vamos a facilitar al usuario una librería de comunicaciones con muy pocas primitivas, de tal manera que la utilización sea sencilla y directa. Estas primitivas servirán como enlace a un *RTS (Run Time System)* residente en cada máquina, encargado de mantener a nivel global, la coherencia de la Memoria Distribuida.

Este *RTS* deberá ser muy parco en su utilización de CPU y recursos del sistema, ya que es el componente de nuestro Sistema Distribuido que estará siempre en ejecución.

Diseño: ¿Cómo lo vamos a hacer?

Como ya hemos dicho, nuestro objetivo es el de proporcionar al usuario, - sea éste un desarrollador u otra aplicación, - un **Modelo Virtual de Memoria Compartida Distribuida**, en forma de un espacio de variables distribuidas.

Principios de Diseño

En este sistema pretendemos recoger y aplicar las experiencias existentes, tanto en la construcción de entornos de progra-

mación distribuida como de la *Ingeniería Software* en general. Debido a ello, en el diseño del sistema hemos considerado tres principios fundamentales: *Principio de Localidad, Predominio de Lecturas* y *Paralelización.*

El primero de ellos, el *Principio de Localidad*, se aplica muy frecuentemente, no sólo en el diseño de programas, sino también en la arquitectura de componentes *hardware*. Según este principio, los datos que han sido utilizados más recientemente tienen más posibilidades de ser vueltos a usar en un futuro que los que hace mucho que no lo han sido. Este principio se puede completar incluyendo en los datos de acceso más probable a los que se encuentran espacialmente cerca de los que ya han sido utilizados.

Esta propiedad ha dado lugar al nacimiento de lo que se denomina caché. Ésta no es más que un sistema de acceso muy rápido, aunque de menor tamaño, que se superpone al sistema normal de datos. Usando las estrategias de utilización adecuadas, y según las hipótesis de localidad, los datos a los que vamos a acceder con más probabilidad se encontrarán en la zona de acceso rápido. Este comportamiento permitirá acelerar globalmente el sistema.

En nuestro sistema vamos a mantener una copia o *caché* de los datos en cada máquina, de tal manera que las consultas sean muy baratas, ya que el tiempo necesario para acceder a una variable local es mucho menor que el de acceder a un sistema remoto.

El segundo principio es el de *Predominio de las Lecturas*, el cual es muy utilizado en las actividades de perfilado, o *profiling*, de sistemas. Este principio se basa en que, en la mayoría de los programas, el número de accesos de lectura es mucho mayor que el número de accesos de escritura.

La proporción lecturas/escrituras depende mucho del tipo de aplicación y de los datos que se manejen. El caso peor se suele dar en dispositivos de almacenamiento masivo, donde pruebas empíricas demuestran que la proporción ronda el 3:1. En las aplicaciones modernas, que incluyen vídeo, audio, etc.., es bastante normal encontrar proporciones de 50:1 e incluso 100:1.

Nosotros vamos a diseñar nuestro sistema para aprovechar esta propiedad, forzando a que todas las lecturas se realicen localmente, lo cual acelerará enormemente el rendimiento del sistema. El precio a pagar es que cualquier escritura es muy cara, al tener que actualizar todas las *cachés* de las demás máquinas.

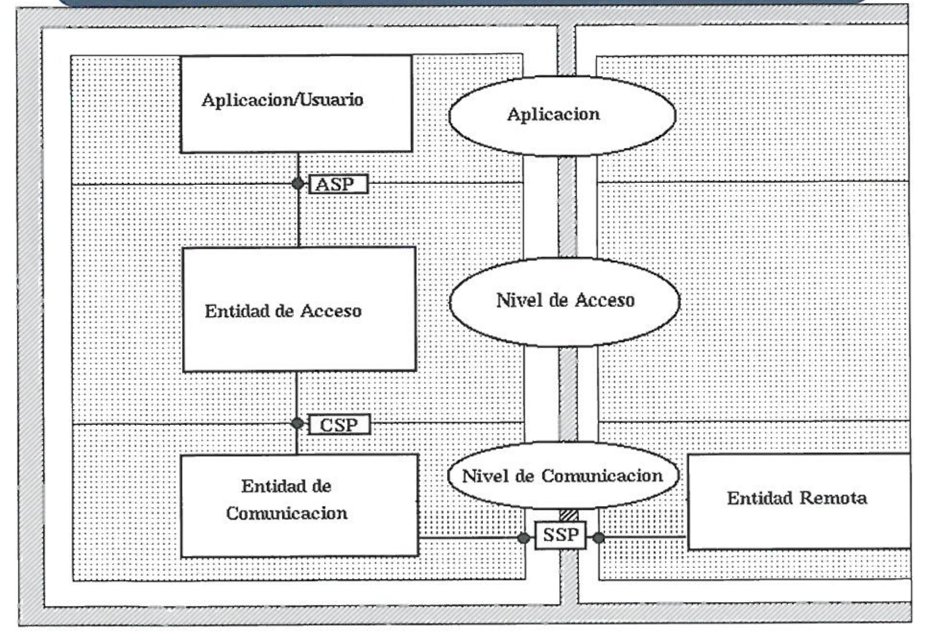
Finalmente, vamos a intentar conseguir un alto grado de *Paralelización*, ya que un alto grado de la misma nos va a permitir aprovechar la presencia de numerosas CPU's en el sistema.

Al realizarse todas las lecturas localmente, el sistema puede estar soportando numerosos accesos de este tipo en sus máquinas, sin que esto suponga un impacto elevado en el uso de las CPU's. El flujo máximo de información aumentará proporcionalmente según la fórmula:

$$Th_{lectura} = \sum Th^{i}_{lectura}$$

Es decir, la capacidad total del sistema se podrá aproximar a la suma de las capacidades de los diferentes nodos de la red. Este tipo de diseño, proporciona una ventaja añadida ya que conseguimos disminuir el tráfico en la red de una manera muy acusada, porque la mayoría de los accesos son locales.

Figura 2.- Descomposición en capa de la arquitectura del Sistema Distribuido. En la figura se representan los puntos de acceso a las distintas capas: SSP, CSP y ASP.



Modelo Estructural del Sistema

Nuestro sistema va a constar de *m* máquinas interconectadas mediante una red de comunicaciones. En cada una de ellas van a coexistir los siguientes elementos:

- n_i **clientes**, usuarios del Sistema Distribuido, comunicándose localmente con un *RTS* único en cada máquina.
- Un **servidor** único por máquina, encargado de proporcionar servicio a los diferentes usuarios, así como de mantener la integridad y coherencia del espacio de datos distribuidos.

Arquitectura

Vamos a construir el sistema modularmente, de manera que añadir aplicaciones sea sencillo y directo. Para ello vamos a construir dos capas o niveles claramente diferenciados. El nivel inferior o *Nivel de Comunicaciones* se encarga de la coordinación y las comunicaciones entre máquinas, gestionando los enlaces entre los diversos *RTS*, mientras que el nivel superior o *Nivel de Acceso* proporciona al usuario una API integrada de programación con las facilidades de Máquina Virtual Única.

Así pues, existen tres interfaces, los cuales interaccionan de la siguiente manera:

- Cuando una aplicación desea acceder a los datos compartidos, envía una ADU[1] al *Nivel de Acceso* a través del punto de acceso correspondiente.
- El *Nivel de Acceso* encapsula la información en una CDU[2], la procesa y la envía al *Nivel de Comunicaciones* mandándola a través de un CSP, es decir, un Punto de Servicio de Comunicaciones.
- El *Nivel de Comunicaciones* local decide si ha de cursar una petición a los demás servidores o bien puede realizar las operaciones el mismo. Si es necesario, se comunica con los servidores remotos a través de los SSP's (Punto de Servicio de Sincronización).

Tabla 1
Primtivas del Nivel de Acceso.

A_START.request
A_START.response

A_CLOSE.request

A_DATA.query
A_DATA.reception

El *Nivel de Acceso* se encarga de aportar una API homogénea a las aplicaciones que utilicen el sistema a través de los diferentes ASP's, para después encapsular los datos y reenviarlos al nivel inferior.

Figura 3.- Ejemplo de transacción de nuestro Sistema Distribuido. En el caso de una escritura se envía la petición a través de las interfases ASP y CSP. A continuación se produce el diálogo entre máquinas a través de la interfaz SSP. Concluido éste se devuelve el control a través de CSP y ASP siguiendo el camino inverso. En el caso de una lectura, nunca llegamos a traspasar la interfaz SSP, realizándose todas las operaciones en la máquina local.

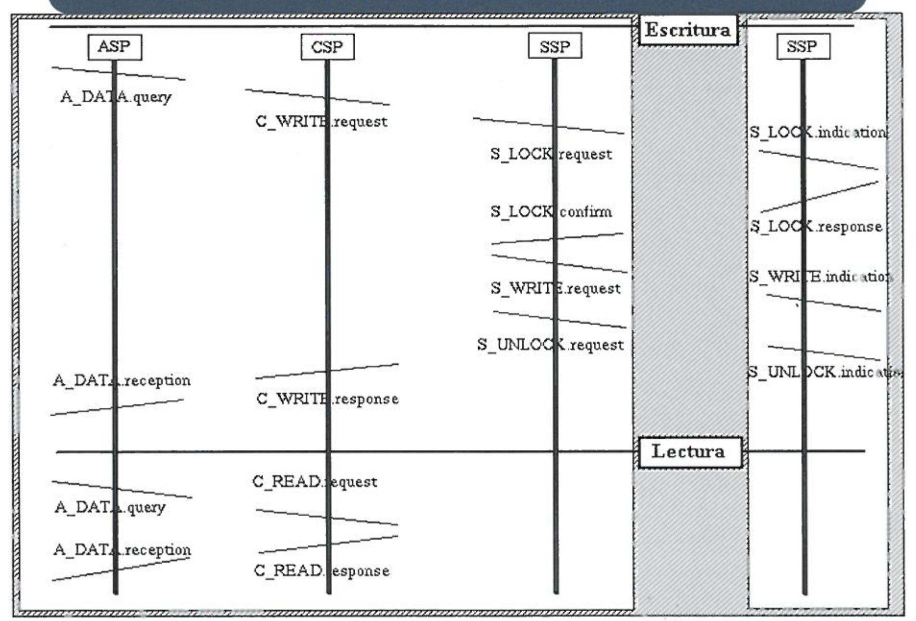


Tabla 2
Primitivas del Interfaz CSP hacia el Nivel de Comunicaciones

C_CONNECT.request
C_CONNECT.response

C_DISCONECT.request

C_READ.request
C_READ.response

C_WRITE.request

Así, cuando una determinada aplicación desea establecer una conexión con el sistema, deberá enviar un mensaje A_START.request as ASP correspondiente. Esta acción es respondida mediante una A_START.response, la cual informa al usuario del establecimiento o fracaso de la conexión.

Una vez tenemos una conexión establecida, el usuario puede mandar mensajes al sistema mediante primitivas A_DATA.query, las cuales generarán respuestas en forma de A_DATA.reception tanto si han tenido éxito como si no.

En cualquier momento, el usuario puede desconectarse del sistema enviando un A_CLOSE.request. Cualquier intento de cerrar una conexión inexistente será ignorado por el sistema.

Tabla 3
Conjunto de las primitivas disponibles para el intercambio de mensajes a través de la Interfaz SSP.

S_LOCK.request
S_LOCK.indication
S_LOCK.confirm
S_LOCK.response

S_UNLOCK.request
S_UNLOCK.indication

S_WRITE.request
S_WRITE.indication

En el *Nivel de Comunicaciones* existen dos interfaces distintas, cada una de las cuales tiene su propio juego de primitivas de comunicación. Los mensajes asociados al interfaz CSP se pueden ver en la Tabla nº2, mientras que los asociados al SSP se ven en la Tabla nº3.

La comunicación a través del Interfaz CSP va a ser tipo *Cliente/Servidor*, en la que todos los intercambios de mensajes se originan en el cliente.

Las primitivas C_CONNECT son encapsuladas directamente sobre las A_START que se reciben, cumpliendo las mismas funciones que aquellas. C_DISCONNECT.request se origina también como traducción directa de A_CLOSE.request. En todos los casos los mensajes estarán anidados, es decir, no se originará una A_CONNECT.response hacia la aplicación hasta que no se produzca una C_CONNECT.response en la interfase CSP.

C_READ.request solicita el valor de uno de los datos de la memoria compartida al servidor, el cual le será entregado mediante una trama C_READ.response. De modo similar, las primitivas del C_WRITE se encargan de realizar las escrituras.

En el caso de que una *Entidad de Comunicaciones* deba contactar con una entidad gemela situada en otra máquina, utilizará para ello la Interfase SSP. Como nuestro diseño solo contempla accesos externos cuando se vayan a realizar escrituras en el sistema, las únicas tramas que nos encontraremos serán las asociadas a las mismas y las asociadas a funciones de sincronización y control de flujo entre las máquinas.

Todas las *Entidades de Comunicación* se encuentran al mismo nivel, pudiéndose originar tráfico en cualquiera de ellas.

Las primitivas de escritura S_WRITE actualizan los valores de la Memoria Distribuida, utilizándose las familias S_LOCK y S_UNLOCK para proporcionar la funcionalidad de transacciones atómicas.

A lo largo de este artículo hemos utilizado los conocimientos que adquirimos en el número anterior para crear la arquitectura de un Sistema Distribuido. A la hora de diseñar esta arquitectura hemos dado un especial énfasis a los criterios de simplicidad, transparencia y heterogeneidad.

Con la arquitectura definida, hemos diseñado un sistema en dos capas, cada una de las cuales se encarga de una tarea específica. Finalmente, hemos establecido las primitivas de comunicación, que permitirán la interacción de los componentes del sistema.

[1] ADU: *Access Data Unit*, mensaje de acceso.

[2] CDU: *Communications Data Unit*. mensaje del nivel de comunicaciones.

Técnica de depuración para el uso de memoria dinámica en C

José Eulalio Poza

La reserva dinámica de memoria permite a los programadores escribir programas en C que van reservando o liberando memoria a medida que la necesitan. El uso dinámico de memoria lleva de forma ineludible a utilizar apuntadores para manejarla.

Mientras los apuntadores se usen de forma adecuada no existe ningún problema, pero en cuanto se cometen errores, se aprecia que no es sencillo localizarlos. A continuación se describen las herramientas que pueden ser utilizadas en C para el manejo dinámico de la memoria, a la vez que se enumeran los errores típicos, y se detalla una técnica para su depuración.

Quizás haya quién piense, que este artículo no esta escrito para él ya que no usa memoria dinámica. Pero a medida que aumenta el tamaño y la complejidad de nuestras aplicaciones llega un momento en el que no queda mas remedio que pasar por este trance y utilizarla. La necesidad de memoria dinámica es patente sobre todo en aplicaciones con estructuras de datos de tamaño variable y dependientes de la entrada del usuario. En estos casos, tenemos siempre dos posibilidades: reservar tanta memoria como sea necesaria para el peor de los casos (si es posible acotarlo), o reservar memoria dinámicamente.

Imaginemos que se desea realizar un programa que trata líneas de texto introducidas por el usuario, pero no se conoce ni el tamaño de cada línea ni el número de líneas que el usuario va a introducir. Dicho de otro modo sólo se sabrá en tiempo de ejecución. Quizás podríamos poner limitaciones al usuario y no permitir mas de *c* caracteres por línea y *l* líneas y reservar un *array* de tamaño *l x c* caracteres. Hay que darse cuenta que de este modo, aunque el usuario tan sólo teclee un carácter, toda esa memoria estará reservada y por tanto no podrá ser utilizada por otros programas o para otro propósito dentro del mismo. En cambio con el uso de memoria dinámica en cada ejecución el programa podría ir reservando memoria a medida que el usuario fuera introduciendo las líneas de texto, optimizando así el uso que se hace de la memoria.

El uso de memoria dinámica es un arma de doble filo y a parte de sus evidentes ventajas y debido a las características del propio lenguaje de programación, resulta muy difícil depurar programas en los que se utilizan equivocadamente los apuntadores a la memoria dinámicamente reservada.

Los errores son difícilmente identificables, ya que en muchos casos dependen de la ejecución y traza del programa e incluso de la coexistencia de otros programas cargados en memoria. Así un programa que aparentemente funciona, no lo hace correctamente cuando otros programas están cargados en memoria o funciona con ciertos datos y no con otros. Esté funcionamiento aparentemente errático puede ser un indicio de un uso inadecuado de la memoria dinámica. Es en este preciso instante cuando comienzan los problemas para el programador ya que la búsqueda del error no es trivial.

Seguramente el error no se encontrará en las últimas líneas de programa que se ejecutaron y quizás al intentar depurar el programa utilizando un debuger el funcionamiento sea distinto.

¿Qué es un apuntador?

El concepto de apuntador es uno de los más confusos para los principiantes, sin embargo un apuntador no es mas que una variable que apunta a otra variable. Dicho de otro modo un apuntador "apunta"al lugar donde se encuentra la información.

Funciones de la biblioteca estándar del C para asignación de memoria dinámica

Las funciones básicas en C son *malloc (memory allocate)* y *calloc (contiguous allocate)* que podríamos denominar como constructoras y la función destructora *free*. A éstas tres funciones básicas hay que añadir la función *realloc (reallocate).*

• Malloc
Se trata de la función más sencilla [Kernighan 1991].

```
void *malloc (size_t n)
```

Esta función retorna un apuntador a una porción de memoria libre de tamaño "n" o NULL si la petición no puede ser satisfecha por el sistema. Es importante reseñar que la memoria reservada tras la ejecución de esta función no estará inicializada.

CONCEPTOS BÁSICOS

Veamos ahora una serie de conceptos básicos sobre los tres elementos que configuran la base donde y para la cual se construirá el gestor de memoria

Sistema Operativo, DOS

El sistema operativo DOS (*Disk Operating System*) se trata de un **sistema monoprogramado y monousuario**, el cual se marca dos objetivos fundamentales como sistema operativo:

1. Conseguir que el funcionamiento interno del ordenador sea transparente al usuario.
2. Permitir la ejecución de programas de alto nivel.

Este sistema operativo esta estructurado en *seis módulos básicos*:

1. **La ROM-BIOS**, se trata de una memoria estática que contiene las primeras instrucciones a ejecutar por el ordenador en el momento mismo de conectarse.
2. **El programa de arranque**, es el encargado de inicializar la carga del sistema operativo cargando en memoria los ficheros IBMBIO.COM y IBMDOS.COM, este programa esta situado en el primer sector del dispositivo de arranque (disquete, disco duro...).
3. **El programa IO.SYS** como su nombre indica se ocupa de interrelacionarse con la BIOS y entre ambos gestionar los periféricos del sistema.
4. **El MSDOS.COM** programa que suministra un interface de alto nivel para los programas de aplicación además de gestionj3ar el sistema de ficheros. Este programa recibe todas las llamadas por medio de la interrupción 21H.
5. **El interprete de comandos COMMAND.COM** programa que queda residente en la memoria y que engloba a todos los *comandos* denominados *internos* (dir, copy, etc.).
6. **El módulo de comandos externos,** son programas no residentes en memoria que están diseñados para la gestión y mantenimiento del ordenador.

Descripción del Hardware

La unidad de proceso como ya se ha mencionado con anterioridad, es el **microprocesador 8086 de Intel**, su elección se debe a ser la base de la familia para la cual se desarrollo el **sistema operativo DOS**. Este procesador de tercera generación posee una arquitectura interna de 16 bits, así que sus registros almacenan datos, instrucciones y operandos cuyo tamaño es de dos bytes, aunque para las transferencias de información del procesador hacia los diferentes periféricos (Input/Output) trabaja con conjuntos de 8 bits, *la característica que más nos interesa es la posibilidad de direccionamiento de este microprocesador.*

Este microprocesador posee 20 bits de direccionamiento lo cual le permite hacer referencia a una dirección entre 1024 x 1024 = 1048576 posibles, quedando la memoria definida desde la dirección 00000H hasta la FFFFFH (en Hexadecimal).

El problema es: **¿Cómo obtener una dirección de 20 bits cuando se trabaja con palabras y registros de 16 bits?**

La solución se denomina **SEGMENTO**. Un segmento es *el espacio máximo de memoria posible a direccionar con 16 bits (64 Kbytes)*. Un segmento puede ubicarse en cualquier posición de la memoria. En un programa todas la direcciones son relativas al principio de un segmento, referenciandose a cada posición dentro del segmento por medio de un desplazamiento representado por 16 bits.

De esta forma cualquier dirección se expresa como una dirección base (segmento) y un desplazamiento a partir de esta dirección.

dirección física = dirección_base*10H + desplazamiento

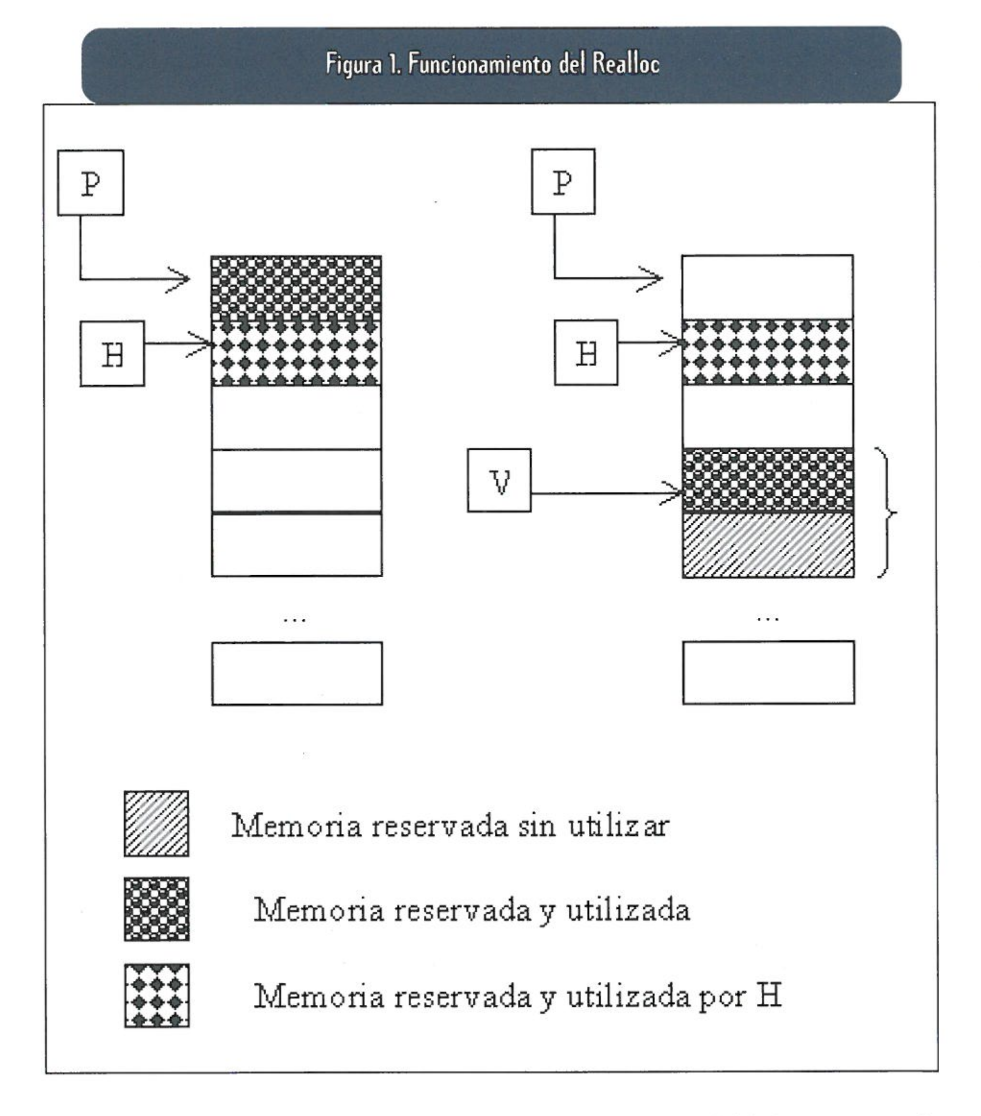
Ejemplo 1:
dirección base o dirección de inicio del segmento = 0506H
dirección de desplazamiento = 1803H
dirección real = 0506H * 10H + 1803H Ë
05060H + 1803HË 06863H

Pero *la expresaremos como*
0506:1803; segmento:desplazamiento

Los Registros

En el interior del microprocesador existen zonas para manipular datos o direcciones y así

Figura 1. Funcionamiento del Realloc



El hecho de que la función retorne un apuntador a *void* (vacío) (también denominado apuntador genérico) posibilita una sencilla conversión de tipos. Evitando así el uso de una operación de *casting* para evitar la no-coincidencia de tipos.

• **Calloc**
void *calloc(size_t n, size_t size)

Esta función [Kernighan 1991] retorna un apuntador a una posición de memoria libre de tamaño *n x size bytes*. O dicho de otro modo un apuntador a un bloque de memoria libre de *n* elementos contiguos, cada uno de los cuales es *de size bytes*. Si el sistema es incapaz de reservar el espacio pedido, la función devuelve *NULL*, debido la mayoría de las veces a la falta de memoria. A diferencia de *malloc*, la función *calloc* inicializa la memoria reservada con ceros.

• **Free**
void free (void *p)

Esta es la función de liberación [Kernighan 1991], mediante ella se podrá liberar cualquier porción de memoria dinámicamente reservada. Es decir, *free* libera el bloque de memoria apuntado por *p*. Recordemos que el hecho de que el tipo del apuntador sea (void *) nos posibilita el que el apuntador sea de cualquier tipo. Si *p* es NULL, la función no realiza ninguna acción. Con esta función sólo podremos liberar memoria reservada dinámicamente.

• **Realloc**
Esta función [Kernighan 1991] es un tanto especial debido a que no reserva ni libera memoria tal y como hemos visto en la definición de las funciones anteriores, sino que cambia el tamaño de la memoria reservada dinámicamente.

void *realloc(void *p, size_t size);

Esta función cambia el tamaño de memoria reservada y apuntada por *p*. Pasando a ser el nuevo tamaño el especificado por el parámetro *size*. Si la demanda se puede satisfacer la función devuelve un apuntador a esta zona de memoria, en otro caso *NULL*.

Al llamar a esta función se pueden presentar los siguientes casos:

- El nuevo tamaño de memoria (*size*) es mayor que el tamaño de la memoria anteriormente apuntada por *p*. En este caso y si el sistema no puede aumentar el tamaño del bloque apuntado por *p* con las posiciones de memoria consecutivas, se verá obligado a buscar un nuevo bloque en una nueva posición en la memoria. A continuación copiará el contenido del bloque apuntado por *p* a la nueva ubicación y liberará dicho bloque.

- El nuevo tamaño de memoria (*size*) es menor que el tamaño de la memoria anteriormente apuntada por *p*. En este caso el bloque de memoria se truncará hasta el nuevo tamaño especificado, y el apuntador *p* no se verá modificado.

Errores más comunes

Una vez definidas las funciones básicas se pasa a enumerar los errores más comunes:

- Memoria no liberada, se utiliza cierta cantidad de memoria dinámica que nunca se libera a pesar de que deja de ser necesaria en el resto del

programa. La causa es o un simple descuido o una liberación erronea. A continuación un ejemplo de liberación erronea. El usuario tras el free, cree haber liberado la memoria pero no es así ya que el apuntador al no poseer el mismo valor que el retornado por la función de reserva, la función free no produce ningún efecto. Nótese que el compilador es incapaz de detectar este error.

```
a *char;
a= malloc(12);
strcpy(a,"ejemplo \n");
a= null; /* a=a++; */
free(a);
```

- Memoria no liberada y no referenciada por ningún apuntador, es un caso particular del caso anterior pero en éste hemos perdido el apuntador a la memoria reservada con lo que será imposible liberarla. Esto puede provocar al igual que en el caso anterior que el programa se quede sin memoria.

- Lecturas o escrituras en posiciones de memoria no reservadas. Suele suceder cuando se libera erróneamente posiciones de memoria que luego vuelven a ser usadas. El error se debe al descuido del programador y también es indetectable por el compilador que no da ningún aviso. En el siguiente ejemplo se reservan 12 bytes y a pesar de que se libera la memoria, vuelve a utilizarse en el *printf*. Este programa es erróneo ya que estamos leyendo de una zona de memoria que no ha sido reservada por lo que su contenido es desconocido. A pesar de ello el programa puede funcionar en algunos casos.

```
a *char;
a= malloc(12);
strcpy(a,"ejemplo \n");
free(A);
printf("Lectura sin reserva:%s ",A);
strcpy(a,"escritura\n");
```

Si el *printf* y el *free* estuvieran separados por cientos de líneas, detectar el

CONCEPTOS BÁSICOS (Continuación)

reducir el tiempo de acceso. Se dispone de 14 de estas posiciones llamadas registros, *son de 16 bits y se reparten en 4 grupos*:

Registros de dirección de segmento

Cuando determinamos la dirección de un segmento Sólo podemos acceder a los 64 Kbytes de memoria contiguos a partir de la dirección marcada por el registro de segmento. Para poder acceder a una mayor cantidad de memoria el 8086 posee **cuatro segmentos**, cada uno de los cuales pose una función determinada.

Registros de segmentos



- **Segmento de Código** (CS), contiene las instrucciones del programa.
- **Segmento de Datos** (DS), almacena variables, constantes....
- **Segmento de Pila** (SS), contiene la dirección de retorno al SO y las diversas direcciones de retorno de las funciones.
- **Segmento Extra** (ES), de uso variado.

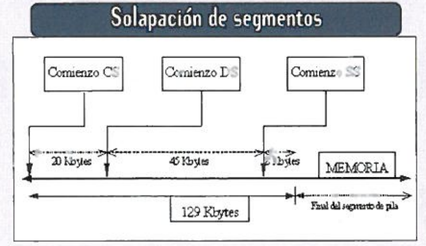
El hecho de que en este sistema de direccionamiento se multiplique por diez (hexadecimal) la dirección base provoca que los 4 bits inferiores sean 0 y por tanto de una dirección posible de segmento al siguiente segmento existirá siempre un diferencia mínima de 16 bytes.

Los segmentos nos dan la posibilidad de acceso a cuatro zonas de memoria de 64 Kbytes cada una, pero los segmentos pueden solaparse quedando la parte inutilizada de un segmento para otro.

Ejemplo 2:

Supongamos que un programa necesita 20 Kbytes para el código, 45 Kbytes para los datos, 8 Kbytes para la pila; y teniendo en cuenta que cada segmento tiene un tamaño fijo de 64 Kbytes, y que no se pudieran solapar tendríamos (64 Kbytes x 3) 192 Kbytes ocupados, sin embargo, permitiendo esta solapación como en la figura podemos colocar los segmentos como sigue y ocupar Sólo 129 Kbytes.

Solapación de segmentos



Registro de flag

Se trata de un único registro de 16 bits que se ocupa de contener información relativa al procesador y al resultado de instrucciones ejecutadas anteriormente, cada bit es un indicador.

Registros de propósito general

Se pueden usar como registros de 8 o 16 bits, en el primero de los casos identificaríamos parte alta y baja.

Registros de propósito general

AX	Registro AH	Acumulador AL
BX	Registro BH	Base BL
CX	Registro CH	Contador CL
DX	Registro DH	Datos DL
	8 bits	8 bits
	16 bits	

- **AX: Registro Acumulador** interviene en las operaciones de Entrada/Salida y aritméticas.
- **BX: Registro Base** es el único que se puede usar en cierto tipo de direccionamientos.
- **CX: Registro Contador** se usa como contador o índice.
- **DX: Registro de Datos** se usa para guardar operadores

suscríbete

a Sólo Programadores

y consigue un **magnífico descuento**

suscripción
normal

ahorro
20%

12 revistas
(1 año)
por sólo...

9.350 ptas.

suscripción
estudiantes
(carreras técnicas)

ahorro
40%

12 revistas
(1 año)
por sólo...

7.050 ptas.







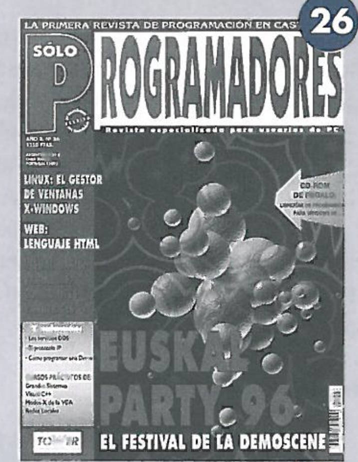


números atrasados

SÓLO PROGRAMADORES

completa ya tu colección





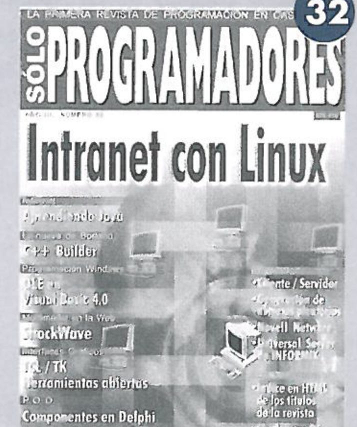
















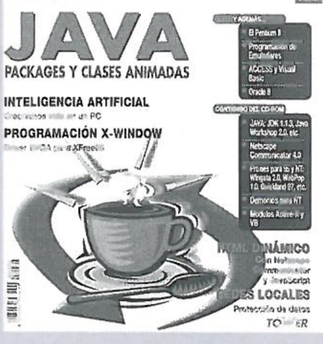












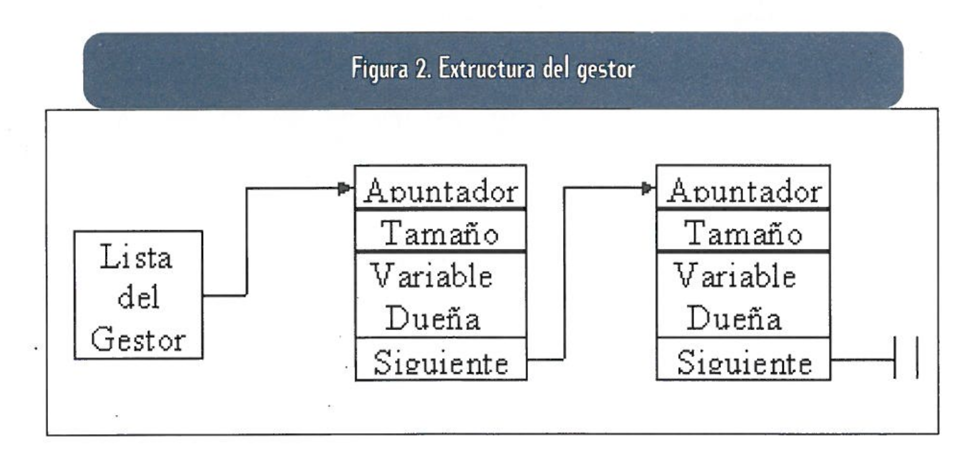
error no sería tan sencillo. A pesar de que el error es evidente el compilador no nos avisa de ello. Pero mucho mas grave es el segundo *strcpy*. Se escribe en una zona de memoria no reservada y por lo tanto no se sabe cuál es su contenido ni si está siendo utilizada. La dificultad en la detección del error dependerá en este caso del sistema sobre el que se ejecute el programa. Imaginad que pasaría si en esas posiciones se encontrara la condición de finalización de un bucle que nada tiene que ver con esta parte del programa. Resulta muy difícil darse cuenta que el bucle no acaba nunca porque a veces un apuntador mal utilizado lleva al programa a escribir en una posición de memoria ocupada por la condición del bucle.

Entonces, si el compilador no resulta de ayuda, el error que se detecta depende de la ejecución concreta del programa y puede tener poco que ver con la causa del error, ¿cómo se pueden detectar este tipo de errores, sin tener que revisar todo el código? Sin duda la mejor solución es no cometer errores, pero como contra eso nadie esta vacunado. Se describe a continuación una técnica que ayuda a la depuración de este tipo de errores.

Una posible solución a la problemática de la programación dinámica

Debido a la dificultad que entraña el hecho de llevar una política correcta de reservas y liberaciones, podemos asaltar la idea de crear un gestor de memoria dinámica. Dicho gestor manejará una lista ligada, en la cual se registrarán todas las operaciones que se realicen. De forma que en cualquier momento de la ejecución se sepa el estado de la memoria dinámica, además de ir creando optativamente un fichero de trazas que nos indique la secuencialidad de las mismas.

Figura 2. Extructura del gestor



Es muy importante tener muy claro los conceptos al desarrollar la idea del gestor, ya que puede parecer un contrasentido el utilizar como herramienta de desarrollo justamente la herramienta, que nos provoca la problemática descrita. Por ello es imprescindible diseñar el gestor sin cometer ningún error.

Estructura del gestor

La funcionalidad del gestor vendrá determinada por los datos que se guarden, la estructura que se propone para el gestor (Fig. 7) consiste en una **lista ligada**; donde *los elementos* de esta lista vienen *definidos como:*

- **Apuntador:** Valor del *apuntador a memoria* el cual hace referencia al bloque reservado.
- **Tamaño:** Valor actualizado en bytes de la memoria solicitada por el usuario al sistema para ese apuntador.
- **Variable dueña:** Cadena de caracteres que identifica al apuntador.
- **Siguiente:** Apuntador al *siguiente bloque de información* de la lista.

Existirá también un valor que representará la *diferencia entre* el número de *peticiones de memoria* atendidas y el número de *liberaciones de memoria* atendidas. Este valor de diferencia ayudará debido a que en una correcta política de reservas y liberaciones la diferencia entre ellas debería tender a cero.

Funciones del gestor

Para no variar los hábitos de los programadores, las funciones que vamos a implementar son versiones de las funciones que el propio C provee y que anteriormente han sido descritas.

Utilizar estas nuevas funciones, descritas en el artículo, proporcionará al programador las siguiente características:

1. Mantener la coherencia de las llamadas a estas funciones, comprobando los valores de los parámetros que se les pasan.
2. Mantener la estructura de datos que hemos descrito anteriormente.
3. Conocimiento continuo del estado de la memoria dinámica, bloques de memoria reservados, tamaño de los bloques, valor del apuntador que realizó la reserva, etc.

Descripción de las nuevas funciones

Mimalloc

Sintaxis:

void *mimalloc (size_t tamanio, char *variable);

Valores de Retorno:

Devuelve un puntero al bloque de memoria recientemente reservado ,o NULL si no existe espacio suficiente para el nuevo bloque. Además de crear el nodo que representará esta operación donde (variable) la cadena de caracteres pasada a la función será el valor de variable dueña. Sólo se producirá la interrupción de la ejecución del programa si se detecta algún problema el cual irá siempre con su correspondiente mensaje de error.

Estructura :

Comprobación de los *parámetros* de la función

Reserva de memoria.

Si *(sistema atiende petición de memoria)*

entonces

Si *(activada estructura del gestor)* **entonces**

Actualizar gestor con la nueva reserva de memoria

Si *(activado el Debug)* **entonces**

damos los posibles **mensajes**

Retornamos el *apuntador* a la primera posición de memoria del bloque recién reservado. o NULL en caso de no poder ser atendida nuestra petición.

Micalloc

Sintaxis:

void *micalloc (size_t num_elementos, size_t tamaño_elemento, char huge *variable);

Valores de Retorno:

Devuelve un puntero al bloque de memoria recientemente reservado de tamaño *num_elemetos x tamaño_elemento bytes*, o NULL si no existe espacio suficiente para el nuevo bloque. Además de crear el nodo que representará esta operación donde (variable) la cadena de caracteres pasada a la función será el valor de variable

CONCEPTOS BÁSICOS (Continuación)

Registros de índices y punteros

Se utilizan para el direccionamiento extendido y para realizar sumas y restas.

Registros de índices y punteros

SP	Apuntador a la Pila
BP	Apuntador Base
SI	Indice Origen
DI	Indice Destino
IP	Apuntador de Instruciones
	16 bits

- **SP: Puntero de pila** (Stack) que va asociado con el registro de pila SS:SP
- **BP: Puntero Base** se usa para acceder a los datos de la pila SS:BP
- **SI: Indice Origen** se usa como índice para las cadenas. DS:SI
- **DI: Indice destino** se usa como índice de cadenas. ES:DI
- **IP: Apuntador a la siguiente instrucción** a ejecutar CS:IP

Los Punteros

Un puntero es una variable, al igual que otro tipo de variable; pero con la peculiar característica de que su contenido es una dirección de memoria. O sea un puntero "apunta" a una dirección de memoria.

Near

Punteros con una longitud de 16 bits por ello no son suficientes para completar los 20 bits que requiere el direccionamiento; así que se ha de recurrir a uno de los registros.

Estos punteros sólo pueden direccionar 64K ya que el inicio del segmento viene fijado por el valor constante del registro de segmento correspondiente. Por esta razón se pueden manipular con total tranquilidad para operaciones aritméticas (como si de enteros se tratara) tanto de modificación como de comparación, sabiendo que nunca se puede superar el tamaño del segmento.

Far

Se trata de un tipo de puntero de 32 bits que se estructura en 16 bits de segmento y 16 bits de desplazamiento dentro del segmento especificado.

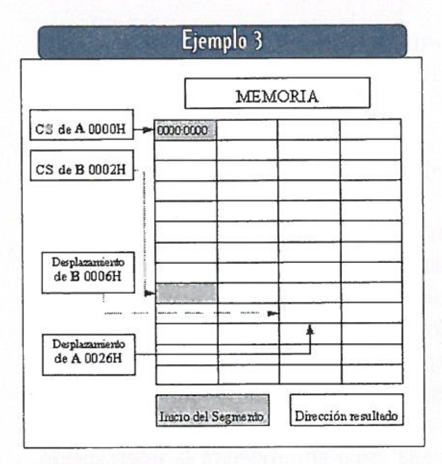
Esta estructura de apuntador permite olvidarnos del registro de segmento y pasar a poseer múltiples valores de segmento y gestionando adecuadamente esta estructura se pueden direccionar más de 64K, por lo tanto poseer programas o datos mayores de 64K.

El inconveniente que surge en este tipo de apuntador es que al existir la posibilidad de varios valores para el inicio del segmento cuando se intentan realizar operaciones aritméticas o de comparación, éstas no son fiables, según se observa en el siguiente ejemplo.

Ejemplo 3:

Sean los apuntadores 0000:0026 y el apuntador 0002:0006 aun teniendo diferentes valores ambos apuntan a la misma posición de memoria

Ejemplo 3



Sin embargo al realizar una comparación nosotros preguntamos si un apuntador es igual a otro. Por lo tanto al realizarse una comparación numérica, se nos retorna que son diferentes.

Otro problema es la imposibilidad de modificar un apuntador FAR aritméticamente ya que este tipo de apuntador no da la posibilidad de modificar el valor del segmento.

Ejemplo 4:

Sea el apuntador de tipo Far con valor: 8906:FFFF si a este puntero le sumamos uno, la

dueña. Sólo se producirá la interrupción de la ejecución del programa si se detecta algún problema el cual irá siempre con su correspondiente mensaje de error.

Estructura :

Comprobación de los *parámetros* de la función

Reserva de memoria.

Si *(sistema atiende petición de memoria)*

entonces

Si *(activada estructura del gestor)* **entonces**

Actualizar gestor con la nueva reserva de memoria

Si *(activado el Debug)* **entonces**

damos los posibles **mensajes**

Retornamos el *apuntador* a la primera posición de memoria del bloque recién reservado siempre que exista sitio en el la memoria libre (far heap) o NULL en caso de no poder ser atendida nuestra petición.

Mirealloc

Sintaxis:

void *mirealloc (void *bloque size_t tamanio, char *variable);

Valores de Retorno:

Devuelve un puntero al bloque de memoria reasignado con el nuevo tamaño (tamanio), o NULL si no existe espacio suficiente para la nueva reasignación del bloque. Además de actualizar el nodo que representará esta operación donde (variable) la cadena de caracteres pasada a la función será el nuevo valor de variable dueña. Sólo se producirá la interrupción de la ejecución del programa si se detecta algún problema el cual irá siempre con su correspondiente mensaje de error.

Estructura :

Comprobación de los *parámetros* de la función

Comprobación del *apuntador* que se pasa como parámetro.

Si *(apuntador es valido)* reasignar memoria (tamaño_en_bytes)

Si *(sistema reasigna memoria)* **entonces**

Si *(activada estructura del gestor)*

entonces Actualizar gestor con la nueva reserva de memoria

Si *(activado el Debug)*

entonces damos los posibles **mensajes**

Retornamos el *apuntador* a la primera posición de memoria del bloque recién reservado. o NULL en caso de no poder ser atendida nuestra petición.

Mifree

Sintaxis:

void mifree (void *bloque, char *variable);

Función:

Libera el bloque de memoria apuntado por "bloque".

Observaciones:

Libera un bloque de memoria previamente asignado con una llamada anterior micalloc, mimalloc, mirealloc, además de liberar su representación en la lista ligada del gestor.

Valores de Retorno:

Ninguno

Estructura :

Comprobación del *apuntador* que se pasa como parámetro.

Si *(apuntador es valido)*

entonces

Liberamos memoria (tamaño_en_bytes)

Si *(activada estructura del gestor)*

entonces Actualizar gestor con la liberación de memoria

Si *(activado el Debug)* **entonces** damos los posibles **mensajes**

sino paramos la ejecución del programa.

Descripción de las funciones complementarias

Durante la explicación en lenguaje pseudo-algorítmico de las funciones que hemos realizado, utilizamos frases que nos dicen lo que debemos hacer, bajemos un nivel más en la especificación:

Veamos ahora las funciones que se usan para completar este módulo de gestión de memoria:

mem_status

void mem_status(void)

Función que muestra por la stderr un resumen del estado actual de la memoria dinámica en este momento de la ejecución.

memoria_del_user

size_t memoria_del_user(void)

Función que nos dice la cantidad de memoria que el usuario a ido solicitando.

creainfobloq

int creainfobloq (void *bloque, size_t tamaño, char *etiqueta)

Esta función crea un nodo del tipo infobloq y la inserta al comienzo de la lista, para luego rellenar los valores con los valores que extraemos de la estructura del sistema:

- apuntador al bloque que estoy mapeando
- tamaño solicitado por el usuario
- etiqueta del nodo, este campo sirve para identificar a cada apuntador, si el usuario le asigna diferentes valores a cada llamada

Retorna:

0 en caso se no haber podido realizar la inserción del nuevo nodo.

<0 si el parámetro pasado es NULL.

>0 en cualquier otro caso.

liberainfobloq

void liberainfobloq(void huge *bloque)

Elimina de la lista el nodo que tiene mapeado al apuntador que se le

pasa a la función como parámetro, en caso de encontrar algún problema se informara al usuario por medio de un mensaje.

leer_infobloq

infobloq *leer_infobloq(void huge *bloque)

Procedimiento que devuelve el INFOBLOQ, que cumple que el valor del campo apuntador. Este es el puntero que se le pasa por parámetro, en caso de no existir el puntero en la estructura se detiene la ejecución y sino esta activa la *estructura* se retornará *NULL*.

Comentario final

Se ha descrito una técnica para la construcción de una herramienta de depuración de errores en el uso de la memoria dinámica en C.

Esta técnica es exportable a cualquier otra plataforma y lenguaje que plantee el mismo problema.

La idea ha sido implementada en i80xxx bajo el sistema operativo DOS y los programas han sido desarrollados bajo los estándares de ANSI C.

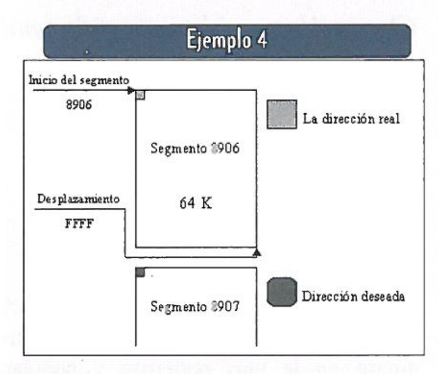
Bibliografía

- [Kernighan 1991]
  "The C programming languaje" de Brian W. Kernighan y Dennis M. Ritchie
- [Wang 1992]
  "An introduction to ANSI C on Unix" de Paul S. Wang
- [Sobelman-Krekelberg 1986]
  "Tecnicas avanzadas en C" de Gerald E. Sobelman y David E. Krekelberg

CONCEPTOS BÁSICOS (Continuación)

idea es que el apuntador pasaría a apuntar a la primera posición del siguiente segmento, es decir, pasaría a valer: 8907:0000, sin embargo, el valor que toma es: 8906:0000; análogamente para el resto de operaciones aritméticas que impliquen cambiar el valor del segmento.

Ejemplo 4



Huge:

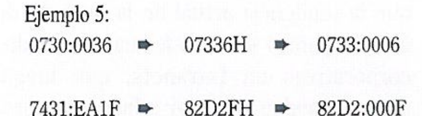
La estructuración y la funcionalidad de los punteros FAR es compartida por los apuntadores Huge.

La diferencia de este tipo de apuntador es una serie de características para paliar los problemas vistos anteriormente con los apuntadores FAR, para conseguir este fin lo que se incluye en este tipo de apuntadores es la definición de normalización.

Un puntero está normalizado cuando el valor que posee el segmento es el máximo posible.

Para normalizar un puntero, se debe pasar a una dirección de 20 bits, entonces los 4 bits de la derecha se usan como desplazamiento dentro del segmento y los 16 bits restantes se usan para la dirección de segmento.

Ejemplo 5:



En el ejemplo anterior, se consigue que si una dirección de memoria es referenciada por dos apuntadores, ambos contengan el mismo valor, de esta manera podremos hacer uso de las operaciones aritméticas sin ningún tipo de problema.

Ejemplo 6:

Operación aritmética (Puntero Normalizado) **retorna** Puntero Normalizado

1803:000F +1 =1804:0000

Modelos de memoria del C para el sistema operativo DOS

Borland C ha sido la herramienta de desarrollo que hemos elegido para el diseño de nuestro gestor de memoria, dentro de esta herramienta veamos lo que realmente nos interesa, el tratamiento que le da a la memoria bajo el sistema operativo DOS. Este entorno impone seis modelos de memoria, los cuales se diferencian en los posibles valores que puedan tomar los registros de segmento y por tanto la cantidad de memoria que se puede direccionar.

Tiny:

Se trata del más pequeño de los modelos de memoria. Se reduce la memoria direccionable a un segmento, para ello los cuatro segmentos apuntan a la misma dirección de memoria.

Small:

La memoria direccionable es de dos segmentos uno para datos y otro para código.

Medium:

Los datos y el Stack (pila) están limitados a los 64K pero el código puede ocupar hasta un Mbyte.

Compact:

Es el espejo del modelo Médium, código limitado a 64K, pero longitud para datos de hasta un Mbyte.

Large:

Se usan para código y para datos con un rango de 1MB.

Huge:

La estructura de memoria que sigue es la misma que el Large pero los accesos a este tipo de memoria son siempre mediante punteros normalizados.

Visual Café, la última apuesta de Symantec

Chema Álvarez (chema@apdo.com)
Ernesto Schmitz (esp@apdo.com)

Sí, parecíamos haber llenado el vaso hasta rebosar, pero Symantec nos ha sorprendido de nuevo, actualizando su gama de entornos de aplicación Java con su nuevo producto: Visual Café for Java. Su facilidad para construir robustas aplicaciones Java con conectividad instantánea a Bases de Datos es enorme.

Introducción

Con el explosivo crecimiento de Internet y el HTML, Java se ha convertido rápidamente en la más poderosa y popular manera de programar en la famosa World Wide Web.

Mucha gente está eligiendo iniciar su recorrido en el mundo de la programación aprendiendo Java como su primer lenguaje. Esto es debido principalmente a que su uso es la manera más fácil de añadir interés y funcionalidad a sus páginas Web. La mayoría de los diseñadores de páginas HTML experimentados recurren a los applets —pequeños programas Java llamados por el código HTML—, para incrementar no solamente la apariencia gráfica de un Web, sino también para implementar canales de comunicación entre la página Web y una base de datos o determinados recursos.

Esto provee una gran mejora en materia de interactividad con el usuario así como potencia a la hora de portar aplicaciones orientadas a la empresa. Y es que la tendencia actual de la mayoría de las compañías es transformar sus redes corporativas en Intranets, que luego, incluso puedan conectar a Internet, aprovechando la infraestructura de ésta para enlazar a su vez las distintas sedes que haya repartidas por todo el mundo.

Además de su relación con HTML, Java se puede usar para crear software independiente y totalmente funcional.

Haciendo historia

Una gran parte del aprendizaje en un lenguaje de programación -como Java—, es aprender a crear código fuente exento de errores. Cualquier error, impedirá al compilador generar el programa. Los fallos de compilación, son el resultado de cosas como errores tipográficos, de puntuación y errores en la sintaxis del lenguaje.

Aprender a programar en un lenguaje significa aprender la manera de situar en el código las instrucciones de forma que el compilador las entienda sin ningún problema. Afortunadamente, Java está bien definido y lógicamente estructurado, de forma que es un lenguaje fácil de aprender.

Cuando Sun Microsystems desarrolló Java, también crearon un compilador para convertir el código fuente en archivos *.class* (archivos de clases). Este compilador se llama javac.exe. Este programa se ejecuta desde la línea de comandos. Para compilar pues un programa Java, tendrías que ejecutar el programa javac.exe pasándole el nombre del código fuente además de una serie de parámetros.

El compilador de Java de Sun, viene incluido en el Java Developers Kit o JDK. El JDK es una colección de herramientas que ayudan en el proceso de compilación, depurado y posterior testeo de programas Java. Al igual que el compilador, el JDK hace uso de la línea de comandos.

El compilador Java proporcionado dentro de Visual Café, fue escrito específicamente para el lenguaje Java y es considerablemente más rápido que el nativo de Sun. Es posible que en algunos casos esta velocidad se multiplique por trece.

Pero la sencillez de este entorno va más allá como nos van acostumbrando la mayoría de herramientas de última remesa. Con Visual Café, en lugar de teclear una serie de comandos, basta con un toque de ratón para compilar o ejecutar un programa.

Este entorno, al igual que otras herramientas aparecidas en el mercado, aunque quizá con menos peso en el desarrollo de software Java, resuelve los obstáculos que podrían existir en la creación de este tipo de programas. Estos obstáculos podrían ser la necesidad de un entorno gráfico apropiado, amigable y visual, así como la disposición de un completo conjunto de librerías heterogéneas, de las que el programador pudiera hacer uso en cualquier momento, eliminando así largos procesos de creación de código de soporte para la aplicación.

Previamente al lanzamiento de este producto y en los inicios de la era Java, una de las primeras compañías que disponían de un completo IDE fue Symantec con una versión especial de su herramienta de C++ modificada especialmente para Java denominada *Espresso*. Precisamente por ser un entorno modificado del existente para otro lenguaje, Espresso hacía que los programadores de Java se viesen obligados a pelearse con elementos del paquete diseñados específicamente para C++.

La única ventaja que ofrecía esto es que si nosotros ya disponíamos de nuestro programa en C++, solo teníamos que transferirlo a Espresso y comenzar a utilizarlo dentro del proyecto.

Posteriormente a este producto, esta compañía remodeló dicha herramienta para adaptarla a los tiempos que corrían, se lanzó entonces Symantec Café, entorno de desarrollo más potente que su predecesor y esta vez, totalmente orientado a Java.

Figura 1. Requerimientos del sistema.

Los requerimientos mínimos del sistema de Visual Café son:
- Intel Compatible 486 ó Superior. (P90 recomendado)
- Windows 95 / NT 4.0 (Workstation o Server)
- 16 Mb para W95 y 32 Mb para NT.
- 30-100 Mb en Disco Duro.
- Monitor SVGA de al menos 800x600.
- Conexión a Internet o a una red Intranet. (Protocolo TCP/IP)

Visual Café

Pero es hora de hablar del presente, y el presente es Visual. Para ello vamos a enumerar y comentar cuales son las principales características de Visual Café, un entorno ideal para aquel que quiera empezar (y continuar) su andadura en el mundo de la programación Java.

Principales características

Antes de nada, vamos a destacar que existen tres versiones de esta herramienta, una denominada *"Web Development Edition"* que equivale a la versión estándar dirigida a la mayoría de usuarios; luego se encuentra la *"Professional Development Edition"* orientada esta vez a programadores de más nivel y por último, la evaluada en este artículo, *"Database Development Edition"*. Todas éstas incluyen el *"Netscape Communicator"* y el *"Visual Page"* (aplicación de Symantec para el diseño de páginas Web). A continuación se enumeran las características comunes a las tres versiones:

- **JDK 1.1** Soporte para JDK 1.1; la nueva versión de Visual Café cumple completamente con las especificaciones del JDK 1.1 de Sun, ya sea el editor de código fuente, los diferentes asistentes, el compilador y el depurador. Además, todas los componentes cumplen con la especificación JavaBeans y están reescritos para soportar el nuevo modelo de eventos.

- **Soporte JavaBeans** La herramienta añade también soporte para JavaBeans (más de 100), que no son sino componentes que pueden ser insertados en nuestras aplicaciones arrastrándolos y soltándolos en el lugar adecuado.

  Las propiedades intrínsecas de JavaBeans desarrollados por terceros son visibles y pueden ser modificadas. La última versión del JFC de JavaSoft está incluida en la librería de componentes.

- **Conversión automática** La versión especial para el desarrollo de bases de datos, puede, automáticamente, convertir un proyecto diseñado para el JDK 1.02 con el actualmente vigente JDK 1.1.

Las siguientes características sólo las incluyen las versiones *"Professional" y "Database"*.

- **Compilación Nativa** La compilación nativa te permite coger tu código fuente Java y compilarlo a código x86 incrementando notablemente la velocidad y flexibilidad.

- **Depuración incremental** Te permite hacer modificaciones de código en tiempo de depuración. El ciclo pues ya no será el clásico depurar-

editar-recompilar, de manera que podremos eliminar un gran número de situaciones en las que solemos perder el hilo de los cambios que íbamos a efectuar.

- **Empaquetado** Visual Café proporciona soporte para los JAR (archivos de empaquetado Java), que permite a los usuarios desarrollar elegantes ficheros ZIP/JAR necesarios para distribuir sus applets o aplicaciones de una manera mucho más cómoda.

- **Compilador Mejorado** Como es lógico, el compilador de la nueva versión es bastante más rápido que la anterior, siendo mucho más notable su eficacia en proyectos complejos.

Por último, nos vamos a extender sobre las capacidades que sólo incorpora la más potente de las tres versiones, la orientada al desarrollo sobre bases de datos, la versión *"Database"*.

Database Development Edition

Esta versión es la más interesante de las tres pues viene con ayudantes que facilitan en gran medida la creación de consultas a bases de datos, guiándole a uno paso a paso a través de todo el proceso, de manera que sólo tengamos que crear referencias a las correspondientes tablas de las bases de datos para tener al momento un formulario en una página Web enlazado con dicha base de datos.

Además nos mostrará el esquema de la base de datos gráficamente, facilitando en gran medida la visión global de la misma, permitiéndonos crear instantáneamente applets, diálogos para las consultas (querys), y aplicaciones con conexiones interactivas a los datos.

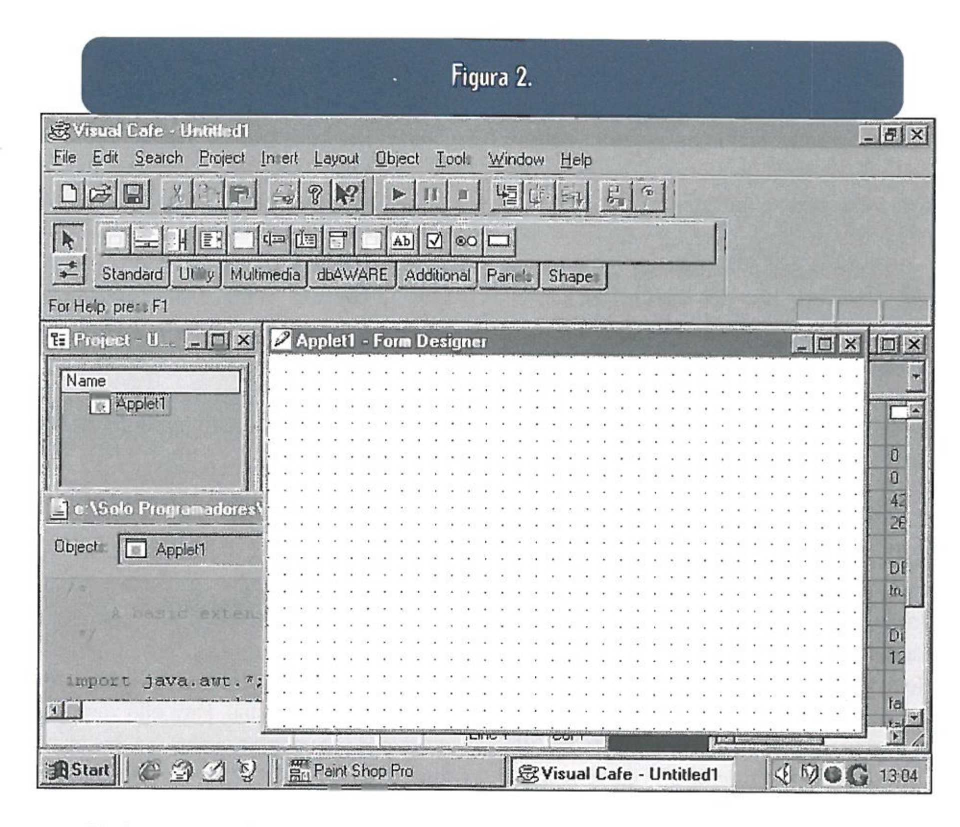


Figura 2.

Podremos elegir de una librería de más de 25 aplicaciones para bases de datos configurables a nuestra medida y más de 100 JavaBeans y Widgets (barras de desplazamiento, texto en movimiento, imágenes animadas, etc.) de las que podremos modificar sus propiedades a nuestro gusto con unas pocas pulsaciones de ratón.

Otra característica enfocada a su uso distribuido son herramientas avanzadas

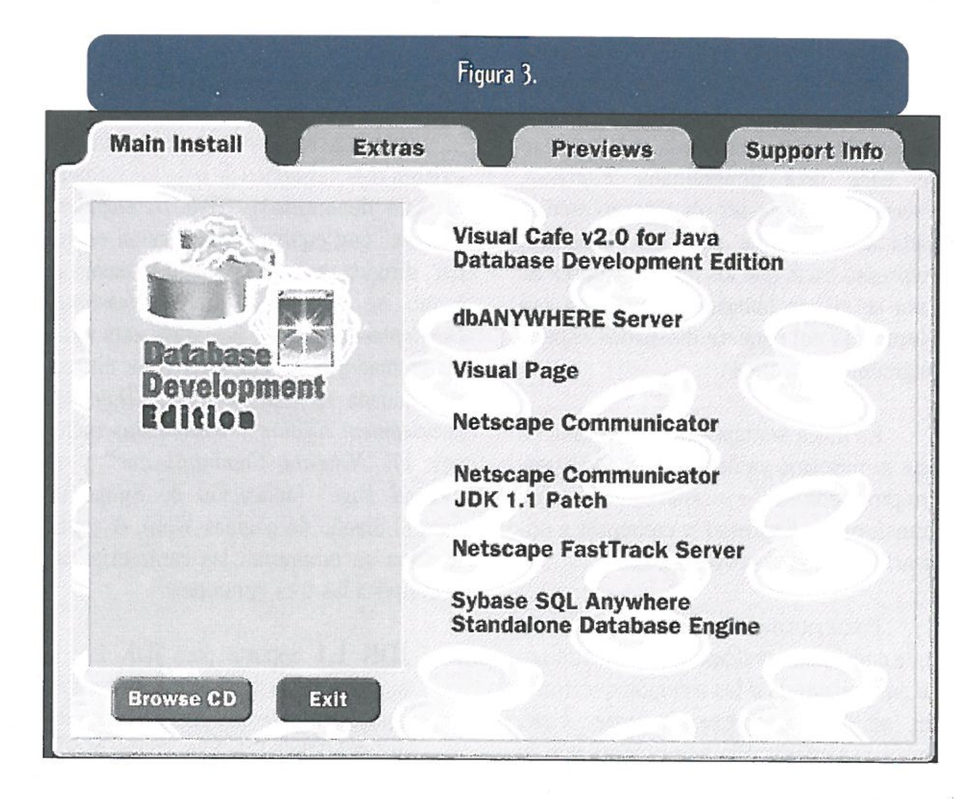


Figura 3.

de depuración como la depuración remota o el depurador incremental que nos permite modificar el código mientras lo depuramos.

Entre el software adicional al ya mencionado incluido en otras versiones está un servidor dbANYWHERE, un Netscape FasTrack Web Server y la base de datos relacional Sybase SQL Anyware. También incluye soporte para JDBC y posee drivers nativos para Oracle, Informix, Sybase y Microsoft SQL Server.

Una vez que las aplicaciones de bases de datos se están ejecutando, los usuarios pueden acceder a los datos a través de cualquier browser que tenga habilitado Java. La arquitectura en forma de árbol jerárquico permite a aplicaciones pequeñas ejecutarse en clientes sin necesitar drivers en el lado del cliente. Además, el servidor aprovecha al máximo el ancho de banda realizando un inteligente sistema de caché de los datos. Todo lo que los clientes necesitan es, como hemos dicho, un browser con soporte Java, por lo que el coste del software de la parte del cliente es muy reducido.

Conclusión

De un tiempo a esta parte, estamos siendo testigos de una típica avalancha de entornos, que aprovechan de la popularidad de un lenguaje para entrar en un mercado de seguro abundante, o lo que es lo mismo, económicamente rentable.

Una cosa no podemos hacer, y es destacar algún entorno en concreto por encima del resto en todos los campos —y no porque no queramos, sino porque es materialmente imposible—. Así pues no, el mejor no es éste, así como no lo es Visual J++, ni Visual Age for Java, ni Java WorkShop ni tampoco la gran mayoría de entornos de terceros.

Así que lo sentimos, pues si lo que el lector desea obtener en la lectura de este artículo es hallar la esencia acerca de si hemos topado ya con el entorno de Java definitivo, pues lo sentimos mucho, pues ese enigma es como el secreto de la composición de la Coca-Cola.

Pero si bien, nada nos haría más felices que recomendar a ciencia cierta un entorno en particular, ya sea por su precio o prestaciones, en este caso estamos situados en una posición un tanto difícil de salir, puesto que la gran mayoría de entornos se encuentran en un nivel muy similar.

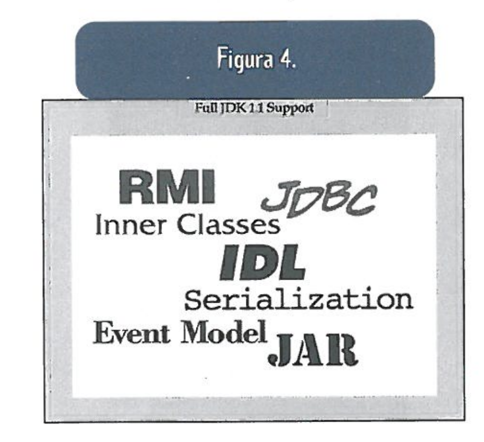
Demos gracias pues, de no tener que realizar una comparativa de los más famosos, ya que nos veríamos obligados a hacer uso de nuestra subjetividad, puesto que a grandes rasgos, o digamos, en el punto de mira del usuario de a pie, cualquiera de ellos cubriría de sobra nuestras necesidades, sean cuales fueren nuestros conocimientos previos o gustos particulares.

Se pueden hacer grandes cosas con este paquete, pues dispone de un entorno muy cómodo de trabajo

Vamos a aclarar esto un poco, de sabida cuenta es la existencia de multitud de usuarios cuya única razón que aportan como interés para aprender este lenguaje, es dotar de más vistosidad a sus HomePages. Pues bien, para ésta gran mayoría están destinadas numerosas aplicaciones que rondan por la Web (muchas de la cuales fueron incluidas en el" Especial de Internet"). Tareas como añadir animaciones, menús, banners, son su objetivo y lo resuelven de una manera muy elegante, y lo que es más, sencilla, puesto que no hay que saber una palabra de Java.

Se pueden hacer grandes cosas con este paquete, pues tenemos todo lo necesario; un entorno cómodo que nos ayuda a realizar ciertas tareas, dejándonos también en el momento que queramos, introducir código directamente, y además, un buen conjunto de librerías, fundamentales a la hora de entrar en campos ya profanados por otros usuarios.

Figura 4.



Si en el primer escalón teníamos esas pequeñas utilidades gráficas para salir del paso, en el último nos encontraremos con el software aportado por IBM y Microsoft, Visual Age for Java y Visual J++ respectivamente, que aunque no exigen del programador unos grandes conocimientos en el lenguaje - al menos en tareas sencillas -, están destinados a cubrir las necesidades más complejas de la industria informática, tales como desarrollo de aplicaciones cliente-servidor y trabajo en grupo sobre recursos compartidos en un mismo proyecto.

En lo más alto

A medio camino de éstas dos, se sitúan herramientas como la evaluada, que nos permiten hacer uso de ellas para alcanzar objetivos de todo tipo, y si es posible —que en este caso lo es—, de la manera más sencilla. Con esto no pretendemos desanimar al usuario más avezado, el mercado de las herramientas de programación en Java está muy pero que muy repartido. Así pues nos vemos obligados a situar un paquete más en lo alto de la pirámide donde se encontraban sendas herramientas de Microsoft e IBM.

Mapas de Mensajes (I)

Xesco Hernández.

Los mapas de mensajes son el mecanismo que permite a las MFC asociar un mensaje Windows con una función miembro de una clase C++. En esta primera entrega veremos cómo funcionan los mapas de mensajes y cómo usar las macros más habituales para escribir funciones que respondan a eventos.

Como es sabido la programación Windows es una programación conducida por eventos. Es decir, un programa permanece en espera y realiza determinadas acciones en respuesta a los eventos que vayan sucediendo. Para asociar eventos con funciones de respuesta las MFC disponen de un mecanismo denominado Mapa de Mensajes implementado en la clase CCmdTarget y en un conjunto de macros.

Visual C++ dispone de Class Wizard para asociar de forma visual y automática un evento o mensaje con una función de respuesta. Class Wizard es capaz de insertar en nuestro código macros de respuesta a mensajes Windows como ON_WM_CLOSE o ON_WM_PAINT, macros de respuesta a notificaciones de controles como ON_EN_CHANGE o ON_BN_CLICKED, y también macros de respuesta a comandos de menú o barras de herramientas como ON_COMMAND. Pero hay una serie de macros para las que Class Wizard no brinda soporte como ON_COMMAND_EX, ON_COMMAND_RANGE, ON_CONTROL_RANGE y ON_MESSAGE. En este artículo veremos cómo funcionan los mapas de mensajes y cómo debemos utilizar dichas macros.

Mensajes

Cuando suceden eventos, como por ejemplo movimientos del ratón, Windows informa a nuestra aplicación del evento sucedido. Windows pasa esta información a través de mensajes que describen el evento, en este caso WM_MOUSEMOVE. En el núcleo de toda aplicación Windows hay un bucle que espera contínuamente mensajes y ejecuta acciones en función de éstos. Todos los mensajes Windows tienen la misma estructura MSG que consiste, básicamente, en una variable "hwnd" del tipo HWND, una variable "message" de tipo UINT, un parámetro "wParam" del tipo WPARAM y un segundo parámetro "lParam" del tipo LPARAM. La variable "hwnd" contiene el handle de la ventana a la cual va dirigido el mensaje. La variable "message" contiene el mensaje en sí mismo (WM_MOUSEMOVE en el ejemplo). Los parámetros wParam y lParam contienen información distinta para cada mensaje en partícular. En el ejemplo del movimiento del ratón contendrían información sobre la posición del mismo.

Esta descripción de los mensajes es propia del API de Windows y por lo tanto es válida para todos los programas Windows, utilicemos un lenguaje u otro, ya que todos han de llamar a las funciones del API.

Para comprender como nuestra aplicación MFC puede responder a mensajes repasaremos primero cómo lo hacen las aplicaciones escritas en C con el API de Windows.

Para crear una ventana con el API primero debemos registrar una clase de ventana Windows con la instrucción RegisterClass() y después crearla llamando a CreateWindow(). Esta clase de ventana Windows está definida por una estructura C llamada WNDCLASS, no tiene nada que ver con las clases C++ (estamos escribiendo código en C). WNDCLASS tiene, entre otros datos, un puntero a función. Esta función es la que responderá a los mensajes dirigidos a todas las ventanas de esta clase. De esta forma, toda ventana tiene asociada una función que procesará todos los mensajes dirigidos a ella, es su WindowProcedure. Nosotros debemos escribir el código de este WindowProcedure y efectuar una u otra acción en función del mensaje recibido.

Las aplicaciones escritas con el API de windows tienen en su función WinMain() un bucle de mensajes, que en su forma más sencilla podria ser:

```
MSG msg;
while(GetMessage(&msg,NULL,NULL,NULL))
        DispatchMessage(&msg);
```

Cuando sucede un evento que afecta a esta aplicación Windows pone un mensaje descriptivo del mismo en la cola de mensajes de la aplicación. El bucle while(), en su llamada a GetMessage() lee el mensaje de la cola. La función DispatchMessage() separa los parámetros que componen el mensaje y llama al Window Procedure de la ventana a la cual va dirigido el mensaje, pasándole los parámetros del mensaje.

En el Window Procedure de nuestra ventana debemos examinar qué mensaje le ha llegado y actuar según lo que queremos que haga el programa. Para ello se suele escribir un switch que tiene distintos cases según el mensaje recibido. En el listado 1 podemos ver un típico Window Procedure de un programa SDK.

A medida que crece la aplicación crece el número de mensajes que se debe procesar y, de este modo, el mantenimiento del switch acaba siendo muy complejo y engorroso.

Esta es la forma en que las aplicaciones C API consiguen responder a mensajes, veamos a continuación cómo resuelven este problema las aplicaciones C++ MFC.

Si las MFC encapsulan el API de Windows todo lo dicho anteriormente para los programas Windows tiene que ser válido para los programas MFC. ¿Entonces dónde están las clases de ventana, los Windows Procedures y el bucle de mensajes?.

Las MFC resuelven el problema de dirigir mensajes a funciones de respuesta utilizando la clase CCmdTarget y los mapas de mensajes. Para que una clase MFC pueda recibir mensajes debe derivar de CCmdTarget. Esta clase contiene toda la funcionalidad necesaria para tratar los mapas de mensajes. En la jerarquía de las MFC podemos ver que de ella derivan entre otras CWnd, CDocument y CWinApp. Los mapas de mensajes son el mecanismo que permite asociar un mensaje Windows con una función miembro de una clase C++. Esta función sólo responderá a un mensaje determinado, evitando así el engorroso switch que necesitábamos en el SDK para distinguir cada mensaje.

Las MFC resuelven el problema de dirigir mensajes a funciones de respuesta utilizando la clase CCmdTarget y los mapas de mensajes

En las MFC también hay Windows Procedures, aunque estas Windows Procedures permanecen ocultos al programador y, desde luego, no tienen la misma importancia que tenían en la programación de SDK. Todas las ventanas creadas en una aplicacion MFC tienen asociado el mismo WindowProcedure, AfxWndProc().

Listado 1

```
LRESULT WINAPI WndProc(HWND
        hwnd, UINT message, WPARAM
        wParam, LPARAM lParam)
{
  LONG ret = 0L;
  switch(message)
  {
   case WM_CREATE:
        RespuestaCreate(hWnd, wParam,
        lParam);
     break;
   case WM_COMMAND:
     switch(wParam)
     {
      case IDM_OPCION1:
        RespuestaOpcion1();
        break;
      case IDM_EXIT:
        DestroyWindow(hwnd)
        break;
     }
     break;
   case WM_DESTROY:
     PostQuitMessage(0);
     break;
   default:
     ret = DefWindowProc(hwnd, messa-
        ge, wParam, lParam);
     break;
  }
  return ret;
}
```

Como en toda aplicación del sistema Windows, hay también un bucle de mensajes que en las MFC está oculto en la función Run() de la clase CWinApp.

Cuando suceden eventos, como clicks de ratón o selección de un comando de menú, Windows pone un mensaje descriptivo del evento en la cola de mensajes de la aplicación. El bucle de mensajes lo lee y llama al WindowProcedure AfxWindowProc() con los parámetros del mensaje recibido. En este punto el mecanismo de mapas de mensajes de las MFC es capaz de resolver qué clase C++ tiene una entrada en su mapa de mensajes y llamar directamente a la función de respuesta indicada en esta entrada.

Listado 2

```
class CMyDlg : public CDialog
{
   ...
   //{{AFX_MSG(CMyDlg)
   afx_msg void OnClose();
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

BEGIN_MESSAGE_MAP(CMyDlg,
        CDialog)
   //{{AFX_MSG_MAP(CMyDlg)
   ON_WM_CLOSE()
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CMyDlg::OnClose()
{
  // TODO: Add your message handler
        code here and/or call default
  CDialog::OnClose();
}
```

Para que una clase de ventana MFC tenga la capacidad de recibir mensajes además de derivar de CCmdTarget debemos escribir la siguiente macro en su declaración:

```
class MyDlg : public CDialog
{
   ...
DECLARE_MESSAGE_MAP()
};
```

En el archivo de implementación de la clase debemos escribir las dos macros siguientes que delimitarán el denominado mapa de mensajes de la clase:

```
BEGIN_MESSAGE_MAP(CMyDlg, CDialog)
...
END_MESSAGE_MAP()
```

En las líneas delimitadas por estas dos macros se irán insertando otras macros que asociarán un determinado evento con su función de respuesta.

Las MFC distinguen tres categorías de mensajes: Mensajes predefinidos de Windows, Mensajes de notificación de controles y Mensajes de comandos, dando a cada uno de ellos un tratamiento distinto.

Mensajes predefinidos de Windows

En esta categoría se incluyen todos los mensajes estándar predefinidos en el API de Windows como WM_PAINT, WM_CLOSE, etc.

Para hacer que nuestra clase CMyDlg responda a un determinado mensaje como WM_CLOSE podemos utilizar ClassWizard. Debemos abrir Class Wizard, ir a la carpeta "Message Maps", seleccionar la clase CMyDlg, seleccionar el mensaje WM_CLOSE y pulsar el botón "Add function".

A medida que asociemos más mensajes a la clase Wizard, esta insertará las macros correspondientes entre estos comentarios

En el listado 2 podemos ver el código que las clase Class Wizard ha añadido a nuestra clase CMyWnd para crear una función de respuesta a WM_CLOSE.

Como se puede apreciar, Class Wizard ha escrito código en tres puntos del programa.

1 Ha añadido una función OnClose a la clase CMyDlg.
2 Ha insertado una macro ON_WM_CLOSE en el mapa de mensajes de CMyDlg.
3 Ha implementado una función de respuesta del tipo OnClose().

Con esto hemos conseguido que la clase CMyWnd tenga una función de respuesta OnClose() que será llamada cada vez que la ventana reciba un mensaje WM_CLOSE.

Los comentarios //{{AFX_MSG y //{{AFX_MSG_MAP son marcas que utiliza Class Wizard para saber en qué punto del código debe escribir. Las líneas de código encerradas entre estos comentarios son mantenidas por Class Wizard, a medida que asociemos más mensajes a esta clase insertará las macros correspondientes entre estos comentarios. El comentario // TODO nos indica que en este punto debemos escribir el código que queremos que se ejecute cuando se modifica el contenido del control de edición.

En la función que ha escrito el Wizard vemos además que después del código que escribamos nosotros llama a CDialog::OnClose(). A menudo la función de respuesta que ya existe en la clase base ejecuta una serie de acciones que son necesarias para el correcto tratamiento del mensaje. Dependiendo del mensaje es necesario llamar a la función de la clase base antes o después de nuestro código o simplemente no hacerlo.

Las MFC definen una macro de mapa de mensajes para cada uno de los mensajes predefinidos de Windows. Estas macros tienen el mismo nombre que el mensaje Windows al cual hacen referencia y le añaden el prefijo ON_. Es decir ON_WM_PAINT para el mensaje WM_PAINT, ON_WM_CLOSE para el mensaje WM_CLOSE. Estas macros no tienen ningún argumento. La función de respuesta que les corresponde está también determinada por las MFC que implementan una función de respuesta para cada uno de los mensajes Windows. Para WM_CLOSE, la función de respuesta debe ser exactamente afx_msg void OnClose().

Cuando una clase como CMyDlg recibe un mensaje WM_CLOSE las MFC buscan en el mapa de mensajes de la clase una entrada ON_WM_CLOSE si la encuentran llaman a la función de respuesta CMyDlg::OnClose(). Si no la encuentran

buscan en el mapa de mensajes de la clase indicada en el segundo parámetro de BEGIN_MESSAGE_MAP(CMyDlg, CDialog). En nuestro ejemplo irían a buscar en el mapa de mensajes de la clase base CDialog.

El efecto de este mecanismo de búsqueda es muy similar al de las funciones virtuales. Si la clase derivada redefine una función virtual se ejecuta ésta, si no se ejecuta la de la clase base. Si el mapa de mensajes de una clase tiene una entrada adecuada, se llama a su función de respuesta, si no se busca en la clase base.

Las clases de las MFC como CDialog y CWnd implementan numerosas funciones miembro de respuesta a mensajes Windows. Cuando en una clase derivada de CWnd como CMyDlg escribimos la función OnClose() lo que estamos haciendo es redefinir una función, que no es virtual, heredada de su clase base CWnd.

En el mecanismo de los mapas de mensajes no hay funciones virtuales. Podríamos pensar que en lugar de mapas de mensajes las MFC podrían haber utilizado funciones virtuales si, como decimos, el efecto es el mismo. El problema sería que las clases bases deberían declarar como virtuales todas las funciones de respuesta. Cada función virtual tiene un puntero en la tabla de funciones virtuales de la clase, vftbl. El gran número de mensajes existentes en Windows comportaría que cualquier clase MFC tendría una vftbl desproporcionada que repercutiría en el tamaño de los ejecutables. Con los mapas de mensajes los programas MFC consiguen unos ejecutables de tamaño aceptable y una respuesta rápida a los mensajes.

Mensajes de notificación de controles

Los controles son ventanas hijas (ventanas con el estilo WS_CHILD) que se encuentran generalmente en el área de cliente de su ventana padre. Cuando el usuario interactúa con ellas, por ejemplo, cuando cambia el contenido de un control de edición o selecciona un elemento de una lista, el control envía a su ventana padre una "notificación" informándole del suceso. Lo hace enviándole un mensaje WM_COMMAND. Este mensaje contiene el identificador del control que lo envía y un código de notificación que especifica qué suceso ha tenido lugar. Por ejemplo, si se modifica el contenido de un control de edición su ventana padre recibirá un mensaje WM_COMMAND y los parámetros wParam, lParam contendrán el identificador del control (IDC_EDIT1) y el código de notificación EN_CHANGE.

Estas macros tienen dos parámetros: un identificador de control y un nombre de la función de respuesta

El API de Windows define un código de notificación para cada uno de los distintos eventos que pueden tener lugar en un control. Todos ellos tienen un prefijo que indica el tipo de control. EN_ para los controles de edición (EN_CHANGE, EN_HSCROLL, EN_KILLFOCUS, EN_SETFOCUS, etc.), LBN_ para las listas (LBN_DBLCLK, LBN_SELCHANGE, etc.), BN_ para los botones (BN_CLICKED, BN_DLBCLK, BN_KILLFOCUS, etc.) y CBN para los combos (CBN_EDITCHANGE, CBN_SELCHANGE, etc.). Las MFC, para cada uno de estos eventos o notificaciones, definen una macro de mapa de mensajes que nos permitirá que una determinada clase responda a la notificación de un control. Estas macros tiene el mismo nombre que la notificación a la cual hacen referencia y le añaden el prefijo ON_. Así tenemos, ON_EN_CHANGE, ON_LBN_DBLCLK, ON_BN_CLICKED, etc.

Listado 3

```
//{{AFX_MSG(CMyDlg)
  afx_msg void OnChangeEdit1();
//{{AFX_MSG

BEGIN_MESSAGE_MAP(CMyDlg,
      CDialog)
  //{{AFX_MSG_MAP(CMyDlg)
      ON_EN_CHANGE(IDC_EDIT1,
      OnChangeEdit1)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CMyDlg::OnChangeEdit1()
{
  // TODO: Add your control notification
      handler code here
}
```

A diferencia de las macros para mensajes predefinidos de Windows, estas macros tienen dos parámetros; identificador del control y nombre de la función de respuesta. Podemos insertar estas macros en el código de una clase de forma automática utilizando Class Wizard. Para ello basta abrir la ventana "Message Maps" del Wizard, elegir el identificador del control y Class Wizard nos mostrará una lista con todos los códigos de notificación que puede generar. Elegimos uno y pulsamos el botón "Add Function". Class Wizzard, a diferencia de lo que ocurría con los mensajes predefinidos de Windows nos propondrá un nombre de función que podremos o no modificar.

En el listado 3 vemos el código que ha escrito Class Wizard para, por ejemplo, la notificación EN_CHANGE de un control de edición con identificador IDC_EDIT1. Al igual que antes Class Wizard ha escrito código en tres puntos.

1 Añade una función en la declaración de la clase.
2 Añade una entrada en el mapa de mensajes de la clase.
3 Implementa una función de respuesta.

Además de éstas, existe en las MFC otra macro para responder a notificaciones de controles que no se puede escribir

Listado 4

```
//{{AFX_MSG(CMyDlg)
...
//{{AFX_MSG
afx_msg void OnChangeEdit1();

BEGIN_MESSAGE_MAP(CMyDlg,
      CDialog)
  //{{AFX_MSG_MAP(CMyDlg)
  ...
  //}}AFX_MSG_MAP
      ON_CONTROL(EN_CHANGE,
      IDC_EDIT1, OnChangeEdit1)
END_MESSAGE_MAP()

void CMyDlg::OnChangeEdit1()
{
}
```

automáticamente con Class Wizard. Dicha macro no es más que la generalización de todas las anteriores. Su sintaxis es la siguiente:

```
ON_CONTROL(ID, Código Notificación, funcion)
```

Tiene el mismo efecto que las anteriores pero, con un parámetro más que especifica a que código de notificación debe responder. Si deseamos usar esta macro deberemos escribirla a mano. Dado que el código que escribamos no será mantenido por Class Wizard debemos hacerlo fuera de sus comentarios.

Las MFC actúan de la misma forma que lo hacen los mensajes predefinidos del propio Windows

En el listado cuatro, vemos como respondemos a la misma notificación que en el ejemplo anterior utilizando ON_CONTROL. Para hacerlo, deberemos escribir código en tres puntos distintos:

1 Declarar la función de respuesta en la clase. OnChangeEdit1().
2 Escribir la macro ON_CONTROL en su mapa de mensajes.
3 Implementar la función OnChangeEdit1().

En lo que respecta al camino que seguirán las MFC cuando necesiten buscar una función de respuesta a un determinado mensaje de notificación, las MFC actúan de la misma forma que lo hacen los mensajes predefinidos del propio Windows.

Si nuestra clase tiene una entrada en el mapa de mensajes para el evento IDC_EDIT1 EN_CHANGE, ejecutan la función llamada OnChangeEdit1(), si nuestra clase no la tiene, buscan en el mapa de mensajes de la clase indicada en el segundo parámetro de BEGIN_MESSAGE_MAP (clase, clase base).

Mensajes de Comandos

La macro ON_COMMAND permite asociar un identificador de opción de menú o de botón de la barra de herramientas con una función de respuesta. Los mensajes procedentes de menús y barras de herramientas se denominan en las MFC comandos.

Si utilizamos class Wizard para escribir una función de respuesta a un determinado comando de menú con identificador ID_OPCION1, deberemos ir a la carpeta "Message Maps", seleccionar el identificador ID_OPCION1 , seleccionar "COMMAND" y pulsar el botón "Add Function". Class Wizard nos pedirá un nombre de función.

Si lo observamos detenidamente, en el listado 5 el código que ha escrito

Tabla 1. Macros de mapas de mensajes.

Macro de mapa de mensajes	Uso	Prototipo de la función de respuesta
ON_WM_XXX	Mensajes predefinidos de Windows	afx_msg void OnXXX()
ON_XXX(ID, funcion)	Notificaciones de controles	afx_msg void funcion()
ON_CONTROL(ID, codigo Notificación, funcion)	Notificaciones de controles	afx_msg void funcion()
ON_COMMAND (ID, OnFuncion)	Comandos (Menú, barra herramientas)	afx_msg void funcion()
ON_UPDATE_ COMMAND_UI(ID, funcion)	Actualización de interfaz de usuario (Menús, barra herramientas)	afx_msg void funcion (CCmdUI* pCmdUI)
ON_COMMAND_ EX(ID, funcion)	Función de respuesta a distintos l ID de comando	afx_msg BOOL funcion (UINT nID)
ON_COMMAND_RANGE (ID1, IDN, funcion)	Función de respuesta a un rango de ID de comandos.	afx_msg void funcion(UINT nID)
ON_UPDATE_COMMAND_ UI_RANGE (ID1, IDN, funcion)	Función de actualización de un rango de ID de comandos.	afx_msg void funcion (CCmdUI* pCmdUI)
ON_CONTROL_RANGE (codigo Notificación, ID1, IDN, funcion)	Función de respuesta a una misma notificación de un rango de controles.	afx_msg void funcion (UINT nID)
ON_MESSAGE (mensaje, funcion)	Mensajes de usuario	afx_msg LONG funcion (UINT wParam, LONG lParam)

Class Wizzard, podremos ver los siguientes cambios:

1 Ha declarado una función void OnOpcion1().
2 Ha escrito una entrada ON_COMMAND en el mapa de mensajes de la clase.
3 Ha implementado una función OnOpcion1().

Llegados a este punto, podríamos pensar que no existe una gran diferencia entre los mensajes de comandos y los mensajes predefinidos de windows. Pero al seleccionar una clase desde Class Wizard podremos observar que las clases que no son ventanas, como por ejemplo un documento, sólo pueden responder a los mensajes de comandos; mientras que las clases de ventanas pueden responder a ambos tipos de mensajes. Que una clase pueda responder a uno u otro tipo de mensajes es debido a que las MFC tratan de forma distinta los mensajes predefinidos de Windows y los mensajes de comandos.

Como ya hemos visto en los mensajes predefinidos de Windows las MFC buscan una función de respuesta en el mapa de mensajes de la clase, y si no la encuentran van a buscar en el de la clase base. Sin embargo, las MFC actúan de manera distinta ante los mensajes de comandos, para buscar una función de respuesta siguen la denominada ruta de comandos.

El primer objeto que recibe un mensaje de comando será siempre una ventana. Así, son las MFC las que redirigen este mensaje hacia otras clases antes de dar a la ventana que lo recibió la oportunidad de procesarlo. En la tabla 2 podemos ver en qué clases se busca una función de respuesta para los mensajes de comandos.

Tabla 2. Ruta de comandos.

Cuando un objeto de este tipo recibe un comando....	Se busca en los siguientes objetos y por este orden una función de respuesta.
Ventana marco de un documento. (CFrameWnd, CMDIChildWnd)	1.Vista activa. 2.Ella misma. 3.Aplicación.
Ventana marco MDI (CMDIFrameWnd)	1.Ventana CMDIChildWnd activa 2.Ella misma. 3.Aplicación.
Vista (CView)	1.Ella misma 2Documento asociado.
Documento (CDocument)	1.El Mismo. 2.Plantilla de documento asociada.
Dialogo (CDialog)	1.El Mismo. 2.Ventana padre del diálogo. 3.Aplicación.

Listado 5.

```
//{{AFX_MSG(CMyView)
  afx_msg void OnOpcion1();
//{{AFX_MSG

BEGIN_MESSAGE_MAP(CMyView,
      CView)
  //{{AFX_MSG_MAP(CMyView)
      ON_COMMAND(ID_OPCION1,
      OnOpcion1)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CMyView::OnOpcion1()
{
  // TODO: Add your command handler
      code here
}
```

Las aplicaciones MFC consiguen dirigir mensajes Windows hacia funciones miembros de clases C++

Como se puede observar en la tabla, no se sigue el orden indicado en la macro BEGIN_MESSAGE_MAP(clase, clase base). Si, por ejemplo, nuestra aplicación tiene una ventana marco CFrameWnd, cuando el usuario seleccione una opción de menú, la ventana marco recibirá el mensaje, pero antes de buscar una función de respuesta en su propio mapa de mensajes, la buscará; primero, en la vista activa, si no la encuentra, en su propio mapa de mensaje, y si no la buscará en la clase aplicación.

Esto permite que los objetos que no son de clases de ventanas puedan responder a comandos. Por ejemplo, en las aplicaciones hechas con AppWizard la función de respuesta al menú FileNew se encuentra en la clase aplicación.

Hemos visto como, a través de los mapas de mensajes, las aplicaciones MFC consiguen dirigir mensajes Windows hacia funciones miembros de clases C++. Las macros que hemos visto explican los usos más habituales de los mapas de mensajes. En la siguiente entrega veremos algunos usos más avanzados de los mismos, que por motivos de espacio no hemos podido incluir en este art;iculo, que nos permitirán; dirigir distintos comandos hacia una misma función de respuesta, utilizar mensajes de usuario o actualizar la interfaz de la aplicación.

Xesco Hernández es programador y actualmente imparte cursos en BIT formación informática. Si el lector desea contactar con él, lo podrá hacer a través de la siguiente dirección de correo : *xesco@redestd.es*

Conectividad Windows-Unix con Samba

Juan José Taboada León, José Juan Mora Pérez

Conectividad es la capacidad que tienen dos o más elementos hardware o software para compartir información, en un ambiente informático. Este término ha pasado de ser de uso exclusivo para las personas relacionadas con el mundo de las comunicaciones, a ser casi un reclamo publicitario.

En los tiempos que corren, lo que importa es que las herramientas con las que trabajamos habitualmente, tengan la capacidad de intercambiar información con cualquier otra herramienta, sea del fabricante que sea.

Ya quedaron atrás los tiempos en los que cada fabricante intentaba imponer su propio estándar, este fue el caso de las comunicaciones, donde aparecieron infinidad de protocolos, tantos o más, como fabricantes. Esta situación dificultaba la capacidad de los elementos informáticos (tanto hardware como software) para comunicarse con elementos de otros fabricantes. Afortunadamente esto ha cambiado, y utilicemos el elemento que utilicemos, ya sea un ordenador, una impresora, o un Sistema Operativo, éste puede ser añadido fácilmente al sistema informático con el que contemos (aunque esto no siempre es cierto), los cuales no tienen porqué ser homogéneos.

Los fabricantes e investigadores han tomado conciencia de los cambios que se están produciendo en la forma de trabajar de los usuarios, y las pautas de desarrollo se centran actualmente, no tanto en desarrollar nuevos elementos para la comunicación, tales como protocolos o elementos HW, sino más bien se buscan nuevos métodos para integrar los elementos existentes, y de esta forma aprovechar los recursos de los que consta una organización. El usuario quiere decidir **donde**, sin importarle ni el **como**, ni el **por qué**. El usuario sólo desea que se realice una comunicación que le permita el intercambio de información entre su ordenador y otros ordenadores, ya sean de su organización o de otra, o compartir ciertos recursos presentes en el sistema informático de su organización.

En este artículo trataremos de hacer accesible los recursos de una máquina Unix a Windows, valiéndonos del servidor Samba y describiendo e ilustrando con ejemplos los pasos necesarios para su instalación y configuración.

SMB, un gran desconocido

Server Message Block es un protocolo definido por Microsoft e Intel en 1987. Es un protocolo del que pocos han oído hablar, pero bastante importante, más de lo que pudiera parecer, ya que el número de usuarios que lo utilizan directa o indirectamente es bastante elevado, al ser la pieza clave del sistema de compartición de recursos, que Microsoft ofrece en sus productos. SMB, se encuentra implementado en sistemas tales como MS-Windows para trabajo en grupo, Windows 95, Windows



Figura 1. SMB dentro del modelo OSI.

NT, en otros como Lan Manager, Pathworks de Digital o en VisionFS de SCO. Como se puede ver en la figura 1, se encuentra situado, dentro del modelo OSI, entre las capas de Aplicación y la de Presentación, por tanto es totalmente independiente del medio de transporte, y esto hace que pueda trabajar sobre TPC/IP, NetBeui o IPX/SPX, figura 2.

La base de este protocolo son bloques de mensajes, a los cuales se le denomina SMBs, y que tanto el servidor como el cliente utilizan para comunicarse. En los mensajes podemos distinguir dos partes, una primera que forma la cabecera del mensaje y cuyo tamaño es fijo. La segunda parte la forman los datos y su longitud es variable. En la tabla 1, podemos contemplar la estructura de un SMB.

SMB ha sufrido diversas modificaciones desde su creación, intentando que cada modificación se ajustara a ciertas necesidades, dando lugar a distintas versiones. Podemos enumerar algunas de estas variantes de SMB como pueden ser COREPROTOCOL, DESCORAZONE, LANMAN1, LANMAN2, NT1, COREPLUS. Podríamos hablar de la existencia de varios protocolos, pero todos guardan un común denominador por lo que son englobados dentro de este grupo y se les denomina dialectos del protocolo SMB. La existencia de todos estos dialectos obliga a que los clientes y los servidores se tengan que poner de acuerdo sobre qué dialecto utilizar, realizando una negociación. El proceso de negociación que realizan cliente y servidor es el siguiente :

- El cliente manda un mensaje *negprot* al servidor y la lista de dialectos que reconoce (SMB_HEADER.Protocol).
- El servidor responde con otro mensaje *negprot* y un número que corresponde con la posición del dialecto que comprende, dentro de las lista facilitada por el cliente. O bien, con un 0xFFFF si el servidor no reconoce ninguno de los dialectos de la lista, rechazando la conexión.

Samba

Samba es un servidor SMB desarrollado por Andrew Tridgell, con la intención de utilizar máquinas Unix como servidores de recursos, tales como ficheros, impresoras y hasta como servidor de Fax (podemos encontrar información de cómo crear un servidor de Fax, en la distribución de Samba). Samba se encuentra implementado sobre una gran variedad de sistemas Unix, tales como Solaris, Linux, UNIXWARE, AIX, NEXTSTEP, NetBSD, SCO Open Server o HP-UX. Esto permite que máquinas Unix que estén presentes en nuestra red sean utilizadas para algo más que como servidores FTP o WWW, conviertiéndose en servidores de ficheros e impresión para clientes SMB. ¿Qué quiere decir esto? Que cualquier persona que esté utilizando alguno de los sistemas que soportan el protocolo SMB, tales como Windows para trabajo en grupo, Windows 95, OS/2, Dos o Windows NT, entre otros, tendrá la posibilidad de acceder (como se verá más adelante, este acceso estará limitado por ciertas restricciones en los servicios) a ficheros e impresoras que se encuentren situados en la máquina Unix, donde se encuentre instalado Samba.

Samba presenta dos ventajas frente a sus (en teoría) competidores (NFS, FTP, NT, VisionFS o Novell), que son :

- Con la instalación de un servidor Samba, se consigue de una forma fácil y barata, que usuarios de sistemas operativos tales como MS-Windows, tengan acceso a ficheros (incluso a nivel de ejecución, si se trata de aplicaciones Dos o Windows) e impresoras situados en una máquina Unix.
- Si lo deseamos, podemos obtener los fuentes del servidor Samba, ya que estos, se encuentran bajo licencia GNU, de esta forma se pueden realizar todas las modificaciones que se crean convenientes para crear un servidor que se ajuste aún más a nuestras posibles necesidades. Una vez que veamos la cantidad de parámetros de los que consta el servidor, comprobaremos que las modificaciones en los fuentes no serán en absoluto necesarias.

Samba es una herramienta en desarrollo y como todo producto en desarrollo también cuenta con desventajas tales como :

- No cuenta con una verdadera asistencia técnica, aunque existen



Figura 2. Modelo de capas SMB.

varios grupos de discusión donde podremos plantear cuantas dudas o problemas se nos presenten.

- El rendimiento del servidor, en cuanto a tiempo se refiere, es inferior al de otras aplicaciones de este tipo, ya que Samba debe avanzar a través de las distintas capas del API de Unix, lo cual incrementa el tiempo de cualquier de las operaciones que se realicen.

- La instalación y configuración del servidor Samba, como veremos más adelante, no es nada sencilla. No debemos confundir la configuración real, que tendremos que realizar en el servidor Samba, con la configuración mínima que necesita el servidor para ejecutarse, ya que podemos poner en funcionamiento el servidor escribiendo un par de líneas en el fichero de configuración (smb.conf). La configuración real, la realizaremos cuando hayamos decidido cuales serán los servicios que debe prestar nuestro servidor, entonces realizaremos una configuración bastante concienzuda para conseguir dos objetivos, por un lado evitar la posible aparición de puntos vulnerables en la seguridad en la máquina Unix, y por el otro, sacarle el máximo partido y rendimiento a los servicios presentes en el servidor Samba.

Servicios

Cuando leemos la documentación dedicada a Samba, encontramos un término sobre el cual gira todo el universo Samba, nos referimos al concepto de *Servicio*. Un servicio es, ante todo, el elemento con el cual se va a enfrentar el usuario que acceda al servidor Samba. Podemos deducir por la definición que hemos dado de Samba, que un servicio pueden ser dos cosas :

- Acceso a cierta parte del árbol de directorios dentro del sistema de ficheros de la máquina Unix, al cual tendremos acceso de lectura/escritura, según se haya configurado el servicio.

- Acceso a alguna de las impresoras conectadas a la máquina Unix, que al fin y al cabo, en Unix, son consideradas parte del sistema de ficheros.

Depuración

Un tema bastante importante, sobre todo durante los procesos de configuración y optimización del Servidor, es el de *nivel de depuración* (Level Debug). Como en cualquier otro servidor, sea del tipo que sea, nosotros como administradores del sistema, debemos conocer cuales son los usos que los usuarios realizan de los recursos disponibles en el servidor, de esta forma podemos diagnosticar posibles problemas o realizar un proceso de refinamiento y optimización del servidor. Samba crea uno o varios ficheros donde se registrará toda la actividad del servidor. La ubicación de estos ficheros se le indica al servidor durante el proceso de compilación y configuración. Por defecto Samba sitúa un fichero smb.log en el directorio "/usr/local/samba/var"

Podemos indicarle al servidor, la cantidad de información que deseamos que genere, para ello utilizaremos o bien un parámetro en el momento en el cual se ejecute el servidor o bien mediante un parámetro en el fichero de configuración. Utilicemos una opción u otra, el valor que tenemos que facilitar al servidor será un entero que va de 0 a 5. El valor por defecto que utiliza el servidor Samba, es 0. Con este nivel el servidor solo nos informará de los errores críticos y las advertencias importantes.

El nivel 1, es el recomendando para un funcionamiento normal del servidor, ya que genera una cantidad pequeña de información, la necesaria para saber como está trabajando.

Los niveles 2 y 3 generan una gran cantidad de datos y deben ser usados solamente cuando existan problemas en el servidor o bien cuando se esté realizando el proceso de ajuste del sistema.

Los niveles 4 y 5 generan demasiada información por lo que estos niveles deberían ser utilizados únicamente por diseñadores y desarrolladores.

Seguridad

Uno de los aspectos más importantes a la hora de implantar un nuevo servicio ya sea www, ftp, telnet, etc. en un servidor, es la seguridad. Cada vez que incluimos un nuevo servicio, éste incrementará la vulnerabilidad de nuestro sistema frente a posibles ataques. Samba no es ninguna excepción a esta regla y podemos citar tres posibles causas del aumento en la probabilidad de un ataque con la instalación de un servidor Samba en nuestro Host. Son las siguientes :

- Las passwords de los usuarios. Por todos, es conocido que uno de los grandes problemas de la seguridad en los servidores son los usuarios, más concretamente las passwords que estos usuarios utilizan. El problema de las password se debe al proceso que se sigue cuando un usuario desea utilizar un servicio del servidor Samba. El proceso de conexión requerirá que se suministren dos password, la primera la solicita Windows para realizar la conexión a red y la segunda la solicita el servidor Samba para acceder al servicio. Si las dos password son iguales, se realizará la petición de la primera, pero no de la segunda, por lo que se podrá acceder al servicio del servidor Samba sin necesidad de introducir una nueva password. Además la passwords no son almacenadas en Windows de una

forma segura, por lo que es recomendable advertir, o si es necesario exigir a los usuarios, que utilicen passwords distintas.

- Una mala planificación de los distintos servicios. Este problema no tiene una fácil solución, ya que no podemos enumerar una serie de reglas para evitarlo. Pero lo que es realmente cierto es que una mala planificación de los servicios del servidor Samba, puede causar bastante daño.

- Una mala gestión de las cuentas de usuario. Partimos del hecho que cada usuario que desee utilizar alguno de los servicios que están disponibles en el servidor Samba, debe tener acceso a una cuenta en el Host, lo cual implica un aumento de la probabilidad de que el sistema sea atacado a través de alguna de estas cuentas.

El acceso a los ficheros del servidor Samba, está controlado, por un lado, por los permisos dentro del sistema de ficheros Unix y por otro lado, por ciertos parámetros de configuración del servidor Samba. La concepción de este sistema no ha sido dejada al azar. Vamos a ver por qué, supongamos que el servidor Samba tiene un servicio como el siguiente :

```
[servicio1]
 path = /etc
 writable = yes
```

Este servicio da acceso al directorio /etc, a todas aquellas personas que tengan una cuenta en la máquina Unix, ya que no se ha indicado una lista de usuarios, además se ha dado permiso de escritura en dicho directorio.

Supongamos que Samba no trabaja de la forma que lo hace, o lo que es lo mismo supongamos que Samba no tuviera en cuenta los permisos del sistema de ficheros del Host. Se nos podría plantear la situación siguiente. Un usuario, el cual tiene acceso a ciertos servi-

Tabla 1. Formato de los Mensajes SMBs

Todos lo mensajes tienen un formato común.

```
typedef unsigned char UCHAR;          // 8 unsigned bits
typedef unsigned short USHORT;        // 16 unsigned bits
typedef unsigned long ULONG;          // 32 unsigned bits
typedef struct {
    ULONG LowPart;
    LONG HighPart;
} LARGE_INTEGER;                      // 64 bits de datos
typedef struct {
    ULONG LowTime;
    LONG HighTime;
} TIME;

typedef struct {
  UCHAR Protocol[4];              // Contains 0xFF,'SMB'
  UCHAR Command;                  // Command code
  union {
    struct {
      UCHAR ErrorClass;           // Error class
      UCHAR Reserved;             // Reserved for future use
      USHORT Error;               // Error code
    } DosError;
    ULONG NtStatus;               // NT-style 32-bit error code
  } Status;
  UCHAR Flags;                    // Flags
  USHORT Flags2;                  // More flags
  union {
    USHORT Pad[6];                // Ensure this section is 12 bytes
    struct {
      USHORT PidHigh;             // High part of PID (NT Create And X)
        struct {
          ULONG  HdrReserved;     // Not used
          USHORT Sid;             // Session ID
          USHORT SequenceNumber;  // Sequence number
        } Connectionless;         // IPX
      }
  };
  USHORT Tid;                     // Tree identifier
  USHORT Pid;                     // Caller's process id
  USHORT Uid;                     // Unauthenticated user id
  USHORT Mid;                     // multiplex id
  UCHAR  WordCount;               // Count of parameter words
  USHORT ParameterWords[ WordCount ];   // The parameter words
  USHORT ByteCount;               // Count of bytes
  UCHAR  Buffer[ ByteCount ];     // The bytes
} SMB_HEADER;
```

cios del servidor, podría acceder al servicio *servicio1*, ya que este aparecería en la lista de servicios disponibles en el servidor, podría seleccionar todos los ficheros que se encuentran en el directorio */etc* y podría pulsar la tecla *Supr*, lo cual produciría el borrado de todos los ficheros de configuración, incluido el fichero passwd del Host. Esta situación dejaría el Host totalmente inutilizado. Para evitar casos como este, Samba debe cumplir las restricciones de los permisos del sistema de ficheros del Host.

Podemos aprovechar este sistema, para combinar los permisos de acceso que tienen los usuarios en el Host, con ciertos parámetros del servidor Samba , tales como *invalid users*, *valid users*, *read list*, *write list*, *writable*, para aumentar la seguridad tanto del servidor Samba como del Host.

Tabla 2. Parámetros de Samba

Aquí están algunos de los parámetros más importantes para configurar el servidor.

- debuglevel, permite especificar la cantidad de información que se generará sobre el fichero smb.log
- log file, nombre del fichero sobre el que se volcará toda la información de depuración.
- root directory, especifica cual será el directorio raíz para el servidor.
- sock address, permite soportar múltiples interfaces virtuales en un mismo servidor.
- workgroup, controla cual será el nombre del grupo de trabajo del servidor.
- available, permite activar/desactivar un servicio.
- allow hosts, lista de hosts que tienen permitido el acceso al servidor.
- copy, copia un servicio determinado.
- guest ok, permite que el acceso a un servicio no necesite password.
- invalid users, lista de usuarios que no tienen acceso al servicio.
- path, camino del directorio perteneciente al servicio.
- postexec, especifica un comando que se ejecutará cuando se termine la conexión a un servicio.
- read only, servicio de solo lectura.
- read list, lista de usuarios que tienen permiso de lectura.
- write list, lista de usuarios con permisos de lectura/escritura.
- comment, comentario del servicio.

Instalar Samba

El proceso de instalación y configuración del servidor Samba es bastante delicado ya que existen una gran cantidad de variantes y cualquier error en algunos de los pasos, podría producir bastantes quebraderos de cabeza. Cuando nos referimos a la palabra error, no sólo queremos hacer referencia a los posibles errores de sintaxis, que pueden producirse durante el proceso de configuración de los distintos servicios y los cuales son fácilmente detectables, sino a los errores semánticos que son los realmente peligrosos. Con estos errores, el servidor funcionará correctamente pero no de la forma deseada, tales como servicios a los que pueden acceder todos los usuarios, o ficheros que pueden ser borrados por cualquiera. Este tipo de problemas son los realmente importantes.

Obtener Samba

Para la obtención del servidor Samba, los más recomendado es que visitemos la página oficial de Samba, en ella podremos encontrar una gran cantidad de información sobre Samba y todos su programas, así como temas de seguridad o los problemas que las distintas versiones de Samba vayan produciendo. Es en esta página donde podremos obtener las últimas versiones disponibles del servidor. A la hora de bajarnos el servidor, debemos decidir entre bajar uno ya compilado para nuestra máquina en particular o por el contrario, bajar los fuentes del servidor para que nosotros mismos podamos compilarlos en nuestro Host. Nosotros hemos probado los dos métodos. Hemos compilado el servidor Samba en un sistema Linux y también hemos bajado un servidor ya compilado (Samba 1.9.16p10) y lo hemos probado en una Sun SPARCstation-4, con SunOS 5.5. En ambos casos Samba ha funcionado correctamente. Para los principiantes en los sistemas Unix, les recomendamos que opten por la opción de bajar un servidor compilado, puesto que de esta forma solo tendrán que dedicarse a las tareas de configuración del mismo, olvidándose del proceso de compilación que puede llegar a ser tedioso.

Esta es la dirección de la página de Samba:

http://samba.anu.edu.au/samba

El programa make

Make es una herramienta que permite automatizar el proceso de construcción de los proyectos. Make necesita un fichero Makefile que le indica cual es la relación entre los distintos ficheros del proyecto. El fichero Makefile está compuesto por reglas las cuales a su vez están compuestas por objetivos, dependencias y ordenes.

```
regla1
objetivo1 : dependencia
            orden
```

Un objetivo puede ser o bien el nombre de un fichero o el nombre de una acción. Cuando ejecutamos el programa make y le facilitamos algún parámetro, este parámetro debe ser el nombre de algunos de los distintos objetivos que for-

man un fichero Makefile. Una dependencia puede ser un fichero u otro objetivo, que se utiliza como entrada para generar el objetivo. Una orden es una acción que make lleva a cabo para realizar el objetivo, puede haber varias ordenes dentro de una regla.

Compilación

Para compilar el servidor Samba en nuestro sistema, tendremos que tener algunas nociones sobre como trabaja el programa make y cual es la estructura del fichero Makefile, ya que será con éste fichero con él que nos vamos a tener que enfrentar. Algunas de las propiedades del servidor las podemos configurar en el fichero Makefile, de esta forma el servidor se encontrará compilado para soportar estas propiedades y no tendremos la necesidad de configurarlas en el fichero de configuración, ya que se tomarán por defecto. Para comenzar el proceso de compilación del Servidor, lo primero es editar el fichero Makefile para poder ajustar algunos parámetros, comprobar la configuración y la existencia de algunos directorios.

El fichero Makefile está compuesto por diversas secciones, todas perfectamente documentadas con líneas de comentarios. Vamos a citar algunas :

```
# Esta línea controla el directorio en el cual se instalarán las páginas del manual.
MANDIR = /usr/local/man

#Directorios dónde se instalarán
#los programas del Servidor.
BASEDIR = /usr/local/samba
BINDIR = $(BASEDIR)/bin
SBINDIR = $(BASEDIR)/bin
LIBDIR = $(BASEDIR)/lib
VARDIR = $(BASEDIR)/var

# Directorios donde se instalarán algunos
#ficheros de auditoría (log)
# y el fichero de configuración (smb.conf).
SMBLOGFILE = $(VARDIR)/log.smb
NMBLOGFILE = $(VARDIR)/log.nmb
CONFIGFILE = $(LIBDIR)/smb.conf
LMHOSTSFILE = $(LIBDIR)/lmhosts
```

Estas secciones son bastante importantes para la instalación del servidor, ya que se decide la ubicación de Samba dentro del sistema de ficheros. Justo debajo de estas secciones, se encuentra una larga lista de sistemas operativos. Están presentes todos los S.O. en los que puede ser instalado Samba. Aquí es donde se le comunica al compilador en qué sistema se está trabajando, esto es necesario ya que no todos los sistemas van a compilar los ficheros fuentes de la misma forma, podemos ver el caso de la sección perteneciente a SunOS, donde aparece un comentario aconsejándonos que compilemos los fuentes con gcc, ya que el compilador C del que dispone este S.O. no es un compilador de C ANSI. Buscamos las líneas correspondientes al sistema operativo en el que estamos trabajando. Habilitamos las dos o tres líneas (según el sistema operativo) que pertenezcan al sistema que estemos utilizando. Supongamos que en nuestra máquina tenemos instalado un sistema Linux sin shadow passsword, entonces buscaremos en el fichero Makefile, la sección correspondiente a este sistema operativo y eliminaremos los caracteres (#) de las filas significativas.

```
# Use this for Linux without
#shadow passwords
# contributed by
#Andrew.Tridgell@anu.edu.au
FLAGSM = -DLINUX
LIBSM =
```

Una vez activadas estas líneas, el fichero Makefile se encuentra a punto para comenzar la compilación e instalación del servidor.

Nuestro fichero Makefile, presenta varias reglas, entre la cuales hay 4 objetivos que podemos utilizar como parámetros con el programa make, son las siguientes :

- **install**, compila e instala los binarios y las paginas para el manual.
- **installbin**, solo compila e instala los archivos binarios.
- **installman**, instala las páginas para el manual.
- **revert**, esta opción es bastante útil cuando decidimos instalar una actualización de Samba, si no borramos la instalación antigua, el proceso de instalación renombra todos los ficheros del servidor, con la extensión .old, de esta forma podemos volver a la instalación anterior, utilizando revert.

Comenzaremos con la opción **install**, y make compilará e instalará todo, tecleando lo siguiente en la línea de comandos :

```
make install
```

En unos minutos y si nada ocurre, tendremos compilados e instalados todos los programas y ficheros necesarios para que se pueda ejecutar el servidor Samba.

Configurar Samba

La configuración de Samba se realiza mediante el fichero **smb.conf.** Este fichero no es creado durante el proceso de instalación por lo que tendremos que hacerlo nosotros mismos. Creamos un fichero que se llame smb.conf y debe estar en el directorio que indica el fichero Makefile, ya que será en este directorio, donde el servidor Samba buscará el fichero de configuración. El fichero de configuración debe de estar, si no se ha modificado su ubicación en el ficheros Makefile, en el directorio: "/usr/local/samba/lib/"

En algunas distribuciones de Samba, hemos probado una que viene ya compilada para Linux, el fichero de configuración, lo situa en el directorio /etc.

Este archivo es la pieza clave del servidor Samba, en el deberán estar des-

Tabla 3. Ejemplo de fichero smb.conf

```
[global]
    workgroup = ALBENIZ-SAMBA
    domain= ALBENIZ-SAMBA
    os level =34
    log file = /usr/local/samba/log/%m
    log level=3

[homes]
    comment = Acceso a ficheros de las cuentas
    path=/export/home/Albeniz/%u

[guest]
    comment = Servicio guest
    path = /usr/local/samba/ficheros/guest
    guest ok =yes
    read only = yes

[servicio1]
    comment = Servicio1 : Servidor Samba SAMBA-19.TAR
    read only = yes
    path = /usr/local/samba/ficheros/samba
    hosts allow = 150.214.163

[servicio_www]
    comment = Servicio2 : Servidor WWW comprimido.
    Valid users = juan,pepe,antonio
    read only = yes
    path = /usr/local/samba/ficheros/www
```

critos, con todo detalle, cada uno de los servicios que nuestro servidor proporcionará. La construcción del archivo de configuración puede llegar a ser tan engorrosa como la complejidad de los servicios que deseamos que el servidor preste.

El fichero se divide en secciones, las cuales quedan delimitadas por el nombre de la sección, que debe estar entre corchetes, y el comienzo de la sección siguiente. La estructura de este fichero recuerda bastante a la estructura de los ficheros INI de MS-Windows.

Las secciones están formadas por parámetros, los cuales son los atributos que definen las características de un servicio determinado. Los parámetros tienen la estructura siguiente :

```
nombre_parámetro = valor
```

Donde nombre_parámetro es el nombre del parámetros y valor es el valor que toma el parámetro. Samba únicamente interpreta un parámetro por línea. Los parámetros se dividen en dos grupos, por un lado están los parámetros globales, los cuales se utilizarán dentro de la sección global (ver más abajo) y afectan a todos los servicios del servidor y por otro lado los parámetros de servicios, que se utilizan en los distintos servicios que se creen para definir sus características. Todos los parámetros de servicios pueden ser utilizados dentro de la sección global, pero los parámetros globales no pueden ser utilizados dentro de las distintas secciones dedicadas a los servicios. En la tabla 2 tenemos una lista con algunos de los parámetros con los que cuenta Samba, más o menos 130. El fichero de configuración tendrá una estructura parecida a la siguiente :

```
[sección1]
 parámetro1 = valor
 parámetro2 = valor

[sección2]
 parámetro4 = valor
 .
 .
 .
```

Hemos dicho que cada una de las secciones que forman el fichero de configuración (smb.conf), describen un servicio determinado, pero esto no es totalmente cierto ya que en este fichero pueden aparecer tres secciones que no describen ningún servicio y son totalmente opcionales :

[global]

En esta sección se describen los parámetros globales del servidor. Esto quiere decir que los parámetros que sean utilizados en esta sección, afectarán a todos los servicios del servidor, siempre y cuando estos mismos parámetros no se encuentren definidos en las distintos servicios. De todos los parámetros existentes, hay unos 60 que pertenecen a esta sección.

[homes]

Cuando el servidor Samba recibe una petición de acceso a un servicio determinado, se busca entre los servicios existentes, si el nombre del servicio solicitado, es encontrado, se utiliza. Si el nombre del servicio solicitado no está en la lista de servicios disponibles, se trata el nombre del servicio solicitado como el nombre de un usuario y se busca en el fichero de passwords. Si el nombre existe y el password es correcto, se crea un servicio nuevo, copiando la sección [homes] del fichero smb.conf y realizando algunas modificaciones:

- Se cambia el nombre del servicio [homes] por el nombre del usuario.

- Si no se ha especificado el parámetro path, se toma como path, el del usuario.

[printers]

Esta sección es exactamente igual que la sección [homes], pero para los servicios de impresión.

En la Tabla 3 podemos ver cual sería la estructura básica de un fichero de configuración. Tenemos 5 secciones, una [global] y otra [homes] ver más arriba y 3 servicios.

En la sección [global] tenemos los siguiente parámetro. *Workgroup* y *domain* que nos permite identificar al servidor SMB desde el exterior. El parámetro *log file* está configurado para que se cree un fichero log por cada máquina que acceda a Samba, con el nombre netbios de la máquina del cliente. Y por último, con el parámetro *log debug*, decidimos un nivel 3 de depuración.

El servicio denominado [guest] es un servicios para invitados, al utilizar el parámetro *guest ok*, ya que Samba no solicitará un password. Podríamos utilizar este servicio como un servicio de ftp.

El siguiente servicio, [servicio1], es de solo lectura y solo se permite el acceso a todos los ordenadores cuyos IP pertenezcan a 150.214.163.*. Esta es una buena herramienta para aumentar la seguridad del servidor.

El servicio [servicio_www], también es de solo lectura, pero en este caso limitamos el acceso mediante el parámetro *valid users*. Sólo la lista de usuarios, de este parámetro, tiene permitido el acceso al servicio.

Test de errores, testparm

Una vez creado el fichero de configuración, es recomendable que comprobemos la existencia de errores en el fichero de configuración. testparm es un programa que realiza este trabajo por nosotros. Si este programa no informa de problemas, devolverá la lista de servicios disponibles en el servidor. Esto quiere decir que no se ha producido ningún error sintáctico y el servidor Samba no tendrá ningún problema para trabajar con este fichero. Pero esto no es una garantía de que los servicios especificados en el fichero de configuración estarán disponibles o desempeñarán la labor para la cual fueron creados.

En la figura tres tenemos un ejemplo de la salida de testparm. Después de utilizar testparm, debemos analizar cada uno de los servicios, para intentar encontrar algún tipo de anomalía en el comportamiento de estos y resolverla si es necesario.

Lanzar Samba

Este es un punto crítico, ya que debemos decidir la forma en la cual se van a ejecutar los programas smbd y nmbd, que son los dos programas clave del Servidor Samba. el programa smbd es el servidor Samba propiamente dicho. Se mantiene a la escucha de las peticiones de conexión que realizan los clientes, esta escucha se realiza por el puerto 139/tcp. El segundo programa, nmbd, es un servidor WINS (Windows Internet Naming Service) que trabaja bajo entornos Unix, este servidor recibe peticiones WINS y se las pasa al mecanismo de resolución de nombres que se encuentre instalado en el servidor Unix.

Una vez que conocemos cual es la función que realizan estos dos programas, vamos a ver cual es la forma en la que

Figura 3. Salida del programa testparm.

```
Load smb config files from /usr/local/samba/lib/smb.conf
Unknown parameter encountered: "domain"
Ignoring unknown parameter "domain"
Processing section "[homes]"
doing parameter comment = Acceso a ficheros de las cuentas
doing parameter path = /export/home/Albeniz/%u
Processing section "[guest]"
doing parameter comment = Servicio guest
doing parameter path = /usr/local/samba/ficheros
doing parameter guest ok = yes
doing parameter read only = yes
Processing section "[servicio1]"
doing parameter comment = Servicio1 : Servidor Samba SAMBA-19.TAR
doing parameter read only = yes
doing parameter path = /usr/local/samba/ficheros/samba
Processing section "[servicio_www]"
doing parameter comment = Servicio2 : Servidor WWW comprimido.
doing parameter read only = yes
doing parameter path = /usr/local/samba/ficheros/www
pm_process() returned Yes
adding IPC service
Loaded services file OK.
WARNING: You have some share names that are longer than 8 chars
These may give errors while browsing or may not be accessible
to some older clients
Press enter to see a dump of your service definitions
Global parameters:
      debuglevel: 3
      syslog: 1
      syslog only: No
      protocol: 5
      security: 0
      printing: 1
      max disk size: 0
      lpq cache time: 10
      encrypt passwords: No
      getwd cache: Yes
```

Tabla 4. Variables de Sustitución

Algunos parámetro del fichero de configuración, permiten el uso del variables de sustitución. Esto permite crear perfiles de usuarios. Podemos habilitar un servicio llamado servicio1 , y nos interesa que cada usuario acceda a su directorio home :

```
[servicio1]
    path = /home/%u
```

Si accede al servicio un usuario cuyo nombre de usuarios es juan, Samba sustituiria la linea path = /home/%u por la línea path = /home/juan

Estas son algunas de la variables que podemos utilizar :

%S = El nombre del servicio actual.
%P = El directorio raiz del servicio actual.
%u = Nombre del usuario del servicio actual.
%G = Nombre del gupo primario de %U
%H = Directorio home del usuario %u
%v = La versión de Samba.
%h = Nombre del servidor sobre el que está corriendo Samba.
%m = Nombre Netbios de la máquina del cliente.
%M = Nombre de la máquina del cliente.
%I = El IP de la maquina del cliente.
%T = Fecha y hora actuales.

deseamos que se ejecuten. Existen dos posibilidades, la primera sería incluirlos dentro de inetd, de esta forma conseguimos que ambos programas trabajen bajo demanda. La otra opción sería lanzarlos en segundo plano (daemos), con lo que ambos programas estarían continuamente esperando peticiones. Se puede deducir, que esta última forma de ejecución será sensiblemente más rápida que la primera. Algo realmente importante, sobre todo durante el proceso de configuración, es que nunca debemos ejecutar ambos programas de las dos formas a la vez, ya que si esto ocurre, Samba puede que no funcione correctamente

Ejecutar Samba desde inetd.conf

Buscamos en el fichero /etc/services, algún otro proceso que esté utilizando el puerto 139/tcp. Este puerto es el que utiliza el servidor Samba por defecto. Si no hay ningún proceso que ocupe dicho puerto entonces añadimos una línea como ésta , siempre que no exista :

```
netbios-ssn    139/tcp
```

Igualmente, para el puerto 137/udp, deberíamos tener una entrada como esta :

```
netbios-ns     137/udp
```

Lo siguiente es editar el fichero de configuración de inetd situado en /etc/inetd.conf y agregar dos líneas :

```
netbios-ssn stream tcp nowait root
    /usr/local/samba/bin/smbd smbd
netbios-ns dgram udp wait root
    /usr/local/samba/bin/nmbd nmbd
```

Volveremos a lanzar **inetd**, mandando la señal HUP .

Ejecutar Samba en segundo plano

Esta opción nos permite lanzar el servidor en segundo plano (daemos), para ellos solo tendremos que ejecutar ambos programas utilizando el parámetro -D, el cual le indicará al servidor Samba, que su ejecución debe ser en segundo plano. Tambien podemos crear un pequeño script.

```
#!/bin/sh
/usr/local/samba/bin/smbd -D
/usr/local/samba/bin/nmbd -D
```

Una vez creado, lo podemos ejecutar desde la misma línea de comandos o bien añadirlo en alguno de los ficheros de arranque del sistema para que el servidor

Tabla 5. Programas del servidor

- smbclient, es un cliente para servidores SMB, que corre en sistemas Unix.
- nmbd, es un servidor WINS (Windows Internet Naming Service) para Unix, recibe peticiones WINS y las pasa al mecanismo de resolución de nombres que se encuentre instalado en el servidor Unix.
- smbd, es el servidor Samba, se mantiene a la escucha de solicitudes de los clientes, en el puerto 139/tcp.
- testparm, este programa chequea el fichero de configuración para comprobar que está perfectamente estructurado.
- testprns, comprueba si el nombre de una impresora en válido.
- smbprint, es un script que nos va a permitir imprimir en una impresora remota.

Samba esté funcionando justo después del arranque del Host.

Pruebas de acceso

Una vez el servidor Samba está lanzado, sería conveniente que realizáramos algunas pruebas, para comprobar que todo funciona correctamente. Estas pruebas las podemos realizar intentando conectar desde una máquina Windows o bien utilizar uno de los programas que vienen con el servidor Samba. **smbclient**, es un cliente para servidores SMB implementado en Unix. Tiene una interfaz que nos recuerda bastante a los programas de ftp de las máquinas Unix. Este programa nos permitirá realizar todas la pruebas que deseemos sobre todos los servicios presentes en el servidor Samba, sin tener que dar acceso al exterior hasta que todo funcione como deseamos. Es conveniente que todos y cada uno de los servicios que presta Samba sea concienzudamente probados con este programa.

Lo primero que vamos a comprobar es que Samba se está ejecutando, para ello, vamos a realizar una solicitud al servidor para que nos devuelva la lista de los servicios disponibles. Ejecutamos la siguiente orden en la línea de comandos.

smbclient -L nombre_hosts

Donde -L es el parámetro con el que le indicamos a smbclient que realice una petición para obtener la lista de servicios disponibles en el servidor SMB, en nuestro caso el servidor Samba, que se encuentra instalado en el host especificado con el parámetro nombre_hosts.

Si queremos tener una lista de todos los servidore SMB, presentes en nuestra red, podemos ejecutar la siguiente linea

Figura 4. Salida del programa smbclient.

```
Connecting to 150.214.163.118 at port 139
Connected
max mux 2
max vcs 1
max raw 65535
capabilities 0x301
Sec mode 512
max xmt 65535
Got 0 byte crypt key
Chose protocol [NT LANMAN 1.0]
Server time is Fri Dec  5 19:05:25 1997
Timezone is UTC-0.0
Domain=[ALBENIZ-SAMBA] OS=[Unix] Server=[Samba 1.9.16p10]
Server gave us a UID of 100. We gave 0
Connected with cnum=73 max_xmit=65531
smb: \> ?
ls              dir             lcd             cd              pwd
get             mget            put             mput            rename
more            mask            del             rm              mkdir
md              rmdir           rd              prompt          recurse
translate       lowercase       print           printmode       queue
qinfo           cancel          stat            quit            q
exit            newer           archive         tar             blocksize
tarmode         setmode         help            ?               !
smb: \> dir

received 7 entries (eos=1 resume=0)
  nueva carpeta                      D         0  Mon Nov 24 17:22:30 1997
  samba                              D         0  Mon Nov 10 19:26:35 1997
  www                                D         0  Mon Nov 10 19:26:42 1997
  En_proceso_de_pruebas                        1  Mon Nov 10 19:26:23 1997
  pstotext.txt                       A      6601  Thu Aug  1 14:32:34 1996
  edit.com                           A     71022  Thu Aug 24 08:50:00 1995
  salud.pas                          A      2935  Sun Mar 23 15:32:16 1997

              61237 blocks of size 4096. 5425 blocks available
smb: \>
```

smbclient -L '*'

Para acceder a alguno de los servicios disponibles en Samba, utilizamos el programa smbclient con los siguientes parámetros :

smbclient '\\HOST\invitados'

Donde HOST es el nombre del servidor al cual se realiza la petición de conexión e invitados es el nombre del servicio al que queremos acceder. El número de caractéres '\' depende de la shell que utilicemos en el Host, a veces es neesario añadir una barra más.

Probemos con el siguiente comando, con el cual podremos acceder al servicio guest.

smbclient '\\albeniz\guest'

Para acceder al servicio1 podemos utilizar la siguiente nomenclatura: servicio, seguido del password, que nos permite un acceso directo.

smbclient '\\albeniz\servicio1' password

Podemos probrar a acceder a un recurso que se encuentre en otra máquina, como puede ser un directorio compartido en MS-Windows, llamado *prueba1* y situado en el ordenador *pcjavier*.

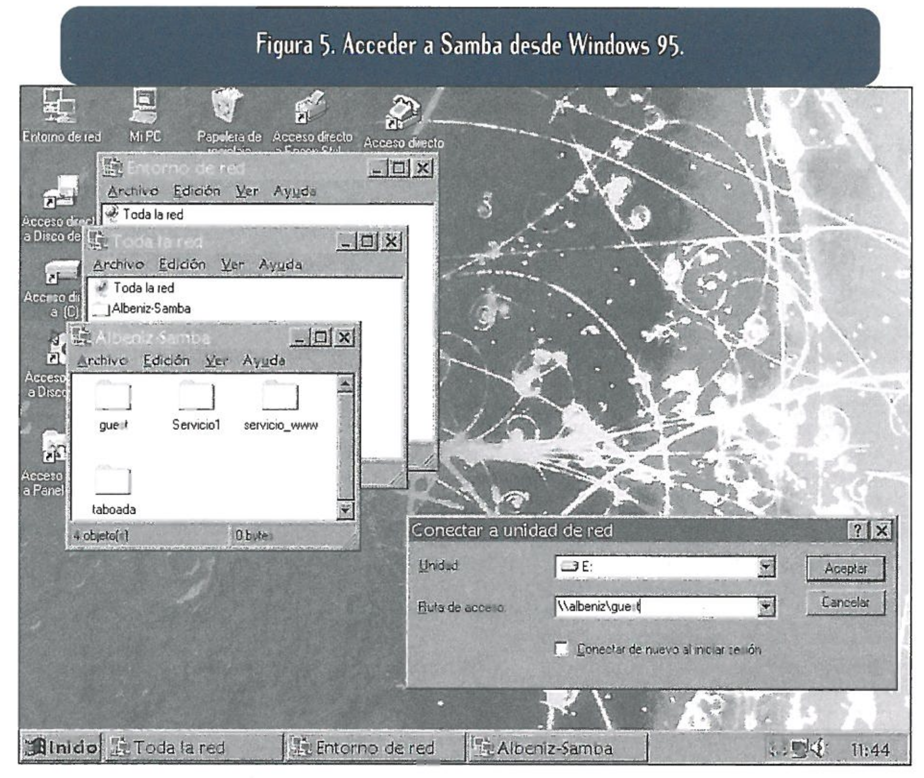
smbclient '\\\pcjavier\\prueba1'

En la figura 4 tenemos un ejemplo de la ejecución de **smbclient**.

Un programa bastante útil, que nos mantendrá informado sobre todas la conexiones abiertas en Samba, es **smbstatus**.

Smbstatus nos facilitará la lista de todas las conexiones a Samba.

Figura 5. Acceder a Samba desde Windows 95.



Acceder desde Windows 95 a Samba

Una vez el servidor está correctamente instalado y configurado, y hayamos realizado las pruebas necesarias para asegurar su correcto funcionamiento, podremos acceder desde cualquier cliente SMB. Vamos a comentar cuales son los pasos para acceder con Windows 95 a un servidor Samba. Lo primero es asegurarnos que tenemos instalados el protocolo TCP/IP necesario para acceder a la máquina Unix. También debemos tener instalado un *cliente de redes Windows*. Tanto el protocolo como el cliente, lo podemos encontrar en propiedades de *Entorno de Red*. Una vez estén perfectamente configuradas las propiedades del sistema, sólo hay que pulsar sobre el icono de *Entorno de Red* y aparecerá un nuevo icono, dentro de la ventana *Entorno de Red*, denominado *Toda la Red*, lo pulsamos y se abrirá una nueva ventana, en ella, a parecen una lista con todos los ordenadores que encontremos en la red. En la lista debe aparecer uno con el nombre del Servidor Samba, este nombre lo hemos introducido como parámetro (workgroup = Nombre_Servidor_Samba) en la sección [global] del fichero smb.conf. Si pulsamos sobre éste, aparecerá una lista de carpetas con los nombres de todos los servicios que ofrece el servidor Samba, figura 5. También podemos ver en esta figura que hay una ventana abierta, con la que podremos realizar una conexión a red.

Puede ocurrir que no aparezca el servidor Samba en la lista, si este es el caso, podemos probar ejecutando la opción *Buscar pc* del icono *Entorno de red*

Acceder desde Dos a Samba

Se puede realizar una conexión a red utilizando el comando net use. Des esta forma asociamos el servicio al que queremos acceder, con una unidad de disco.

net use unidad: \\servidor\directorio contraseña

Por ejemplo: net use g: \\Albeniz\guest

Así disponemos de una nueva unidad lógica que hace referencia al árbol de directorios de Unix en el cual tenemos derechos.

Conclusión

En aquellos artículos que tratan sobre Samba, se comienza valorando muy positivamente sus cualidades, sobre todo su gratuidad y utilidad, pero cuando se hace referencia a los procesos de instalación y configuración, todos los autores manifiestan lo engorroso y complejo de estos procesos, debido al alto número de parámetros del fichero de configuración y a la falta de una interfaz. El lector se va ilusionando y cuando llega a este punto se desvanece todo intento de poner manos a la obra.

Nos pareció interesante la idea y nos pudo más los positivo de Samba. Luego, nos dimos cuenta que tampoco era para tanto y que incluso el hecho de disponer de un número elevado de parámetros nos ofrecía una flexibilidad digna de valorar, al fin y al cabo todo se resume a añadir algunas líneas en el fichero de configuración, a medida que se desee una mayor sintonización del servidor se van añadiendo más parámetros. Además, se dispone de herramientas para ir comprobando si los pasos se han realizado satisfactoriamente.

En este articulo hemos introducido al lector en el protocolo SMB y en Samba, y se ha detallado todo el proceso instalación-configuración de Samba, esperando que permita a otros lectores dedicar el menor tiempo posible en estas tareas.

Por último, agradecer a Andrew Tridgell la creación de Samba. Para cualquier tipo de consulta o sugerencia, se puede poner en contacto con los autores, vía e-mail, en albeniz@uhu.es.

Merced, el futuro ya llega

Jorge Figueroa

Recién salido el Pentium II en el mercado, parecía que no se hablaría de nuevas CPU's hasta bastante más adelante, pero la tecnología evoluciona a una velocidad de vértigo y ahora ya se está empezando a hablar en todo el mundo del nuevo chip llamado P7, conocido con el nombre clave de Merced.

Los microprocesadores Intel actuales x86 basados en la tecnología CISC parece que están llegando a un estancamiento en cuanto a posibilidades de aumentar su rendimiento se refiere. Por esa razón, Intel no quiere que pueda aparecer alguien que le arrebate su puesto número uno, en cuanto a ventas de microprocesadores se refiere (claro está) y que su plataforma vaya quedando en un segundo plano.

La solución para renovar una tecnología saturada es simple y única, sustituirla por otra con más posibilidades de futuro. En principio esto parece muy simple, y mas cuando hablamos de Intel, la empresa que más práctica tiene en el desarrollo de nuevos microprocesadores y nuevas tecnologías. El único problema, que resulta de realizar dicha renovación, es que si se sustituye completamente una pieza tan importante de una plataforma como es el microprocesador (se cambia el set de instrucciones), el resultado es que todo el soft disponible del mercado queda obsoleto al instante, lo cual, es equivalente a abrir una puerta a los posibles competidores (plataformas de otras marcas) para que puedan aprovechar el momento de renovación tecnológica hardware para introducir sus respectivas plataformas mediante mil y una técnicas de marketing.

Un ejemplo de plataforma, que podría ganar terreno, por decir alguna, es la del POWER PC de IBM, que precisamente no ha tenido mucha aceptación por la gran estabilidad de la plataforma Intel x86 y la gran cantidad de soft disponible.

El proyecto merced es fruto de la alianza Intel/Hewlett Packard

Como solución al posible desastre económico, que se podría producir a causa de dicho renovamiento, Intel ha encontrado la mejor solución que podía idear. Para empezar, antes de nada se ha aliado con otra empresa importante y conocida por casi todos, Hewlett-Packard, para cooperar en el diseño de un nuevo microprocesador llamado de momento Merced (P7) el cual se caracterizará principalmente por que poseerá capacidad para ejecutar dos sets de instrucciones (dos microprocesadores en uno sólo, por decirlo así), el set x86 por un lado y el IA-64 por otro.

Con Merced se sustituirá la tecnología CISC sin provocar molestias al usuario final

Diseñando un chip así, no se pierde el hilo de la tecnología existente en el mercado y se puede ir introduciendo poco a poco el soft para el segundo set de instrucciones (IA-64), que es el que supondrá propiamente dicho una novedad tecnológica, tanto por el diseño RISC de la CPU como por estar diseñado con 64 bits puros (más gigabytes direccionables). Tras esta introducción, pasaremos a explicar todo lo que más o menos se sabe de este nuevo chip en todos sus aspectos.

Orígenes de la tecnología RISC

Los primeros vestigios de la tecnología RISC (Reduced Instruction Set Computer) los encontramos en los viejos 801 de la casa IBM, el cual apareció a partir de la idea de crear un microprocesador que no tuviese mucha complejidad lógica interna (de arquitectura), lo cual se consiguió a partir de crear un set de instrucciones, que usará un amplio set de registros en vez de usar muchos modos complejos de direccionamiento y que a su vez fuesen todas lo bastante simples como para poder ejecutarse en un sólo ciclo de reloj del oscilador. La Directiva (premisa) de la cual partía todo este planteamiento de diseño era simple: traspasar la complejidad de la CPU (Instrucciones) a los propios compiladores que operan sobre ella.

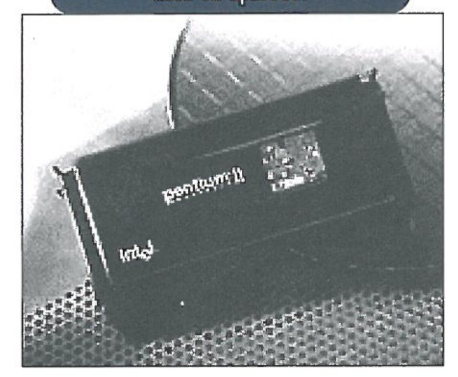
Merced puede ejecutar tanto instrucciones x86 como IA-64

Aunque actualmente muchas personas consideran al Merced como un simple chip más del tipo VLIW (Very Long Instruction Word), habrá muchas diferencias esenciales en su interior. Los micros actuales de este tipo usan mucha lógica interna compleja para la detección de secuencias de instrucciones que puedan ejecutarse en paralelo, en cambio, en el caso de la tecnología IA-64, aunque básicamente podemos decir que es igual en cuanto a concepto, esta lógica interna no existirá si no tan sólo a muy baja escala, ya que esta tarea la deberá poder precalcular el propio compilador a la hora de construir las secuencias de instrucciones de los fuentes binarios que se generen para que queden ordenadas de la mejor forma para poder ser detectadas de forma sencilla para la paralelización con la lógica elemental destinada a ello. En la práctica, ello simplifica enormemente el diseño del chip y facilita crear un diseño más rápido y con menos cuellos de botella de procesado.

TABLA1: Evolución en el tiempo de los microprocesadores Intel:

*-**3000** , 2N bits, fabricado en TTL-Schottky. Uno de los primeros microprocesadores del mundo.
*-**4004** , 4 bits (usado en calculadoras).
*-**4040** , 4 bits.
*-**8008** , 8 bits, fabricado en PMOS.
*-**8021** , 8 bits, fabricado en NMOS.
*-**8035** , 8 bits.
*-**8039** , 8 bits.
*-**8041** , 8 bits.
*-**8048** , 8 bits.
*-**8049** , 8 bits.
*-**8080** , 8 bits, fabricado en NMOS.
*-**8080A** , 8 bits, fabricado en NMOS.
*-**8085** , 8 bits, hasta 3 Mhz. Fabricado en NMOS.
*-**8086** , 16 bits, hasta 12 Mhz. Fabricado en NMOS. Apareció en el mercado en 1978.
*-**8741** , 8 bits (incluye REPROM).
*-**8748** , 8 bits (incluye REPROM).
*-**8085A-2** , 8 bits , 5 Mhz. Fabricado en NMOS.
*-**8088** , 16 bits, llega hasta 12 Mhz.. (similar al 8086).
*-**8087** , 16 bits (coprocesador 8086).
*-**80186** , 16 bits, hasta 16 Mhz.
*-**80286** , 16 bits, llegaba hasta los 20 Mhz.
*-**80287** , 16 bits, coprocesador del 80286.
*-**80386** , 32 bits, poseía velocidades de hasta 40 Mhz. Modelos SX, DX.
*-**80387** , 32 bits, coprocesador del 80386.
*-**80486** ,32 bits,máx.100 Mhz. Modelos muy diversos, DX, DX2, DX4, SX y SL, el primer modelo apareció en 1989.
*-**Pentium** , 32 bits , hasta 200 Mhz. Fabricado en BiCMOS. Apareció en el mercado en 1993.
*-**Pentium Overdrive** (Ampliación para placas 486 a Pentium).
*-**Pentium MMX**, hasta 233 Mhz, doble caché que su antecesor. Aparecido en el mercado en 1996.
*-**Pentium Pro** , 32 bits bits + Extensión, 16 kb de caché, tecnología MMX también integrada. Llega hasta 200 Mhz. Capacidad para poder gestionar hasta 64 Gbytes en un modo especial de funcionamiento. Este micro ha sido introducido en el mercado a finales de 1996.
*-**Pentium II**, 32 + Extensión , velocidades de 200-233-266 y 300 Mhz por el momento, 5.5 millones de transistores. Introducido en el mercado en 1997.
*-**Merced**, 64 bits, poseerá velocidades iniciales mínimas supuestas de 250 Mhz, 8 millones de transistores, 2 sets de instrucciones integrados. Tecnología MMX también incluida. La fecha, no final, es que se empezará a vender a partir de 1998-1999. Habrá tres versiones iniciales diferentes del micro para diversos campos de la informática (Merced, Tahoe y otro desconocido).

La reciente aparición del Pentium II asegura que Merced tardará algunos años en aparecer



La caché del microprocesador

Los microprocesadores actuales están limitados a usar predicción de datos mediante la implementación de lógica inteligente en el mismo micro ya que el set de instrucciones actual le hace muy difícil a los compiladores poder cooperar con la CPU en la predicción de datos que se van a usar en cada momento a continuación.

Sabido este problema de la tecnología actual, IA-64 no va a repetir el error y se incluirá todo lo necesario para poder solventar este problema.

Por qué de nuevo RISC

Como ya se ha dicho, la idea de crear un microprocesador RISC no es nueva, ya que en los años 80 ya se crearon micros basados en arquitectura de este tipo pero que no llegaron a alcanzar niveles de rendimiento muy elevados, debido principalmente, a que en esos años los chips que se fabricaban estaban hechos a una escala de integración muy baja por limitaciones tecnológicas de la época.

La causa de que quedaran en el olvido posteriormente hasta la actualidad, vino dada por que debido a su simplificación de diseño lógico interno, pasaron todo el trabajo de optimización de rendimiento a los compiladores que funcionaban sobre ellos, lo cual hacía que para poder realizar compilaciones de los fuentes , se requirieran ordenadores 100 veces más potentes para poder realizarlas con una mínima velocidad (un mínimo rendimiento del sistema para el desarrollo de aplicaciones) debido a la gran cantidad de cálculos que tenían que realizar para escribir los binarios resultantes de los fuentes.

Merced es un microprocesador RISC

Dada y conocida esta premisa del pasado, está claro que la tecnología IA-64 no poseerá esta lacra, y debido a su modernidad de diseño, requerirá como mucho los requerimientos de potencia que se necesitaban para los viejos Cyberdrome (un antiguo modelo de máquina)

Unidades de procesado

Hay fuentes que afirman que los micros IA-64 tendrán cuatro unidades de procesamiento, pero dichas unidades no estarán diseñadas con sistemas lógicos de detección automática de concurrencias entre instrucciones x86 como posee por ejemplo el Pentium Pro, si no que se definirán nuevas normas (directivas) para los nuevos compiladores nativos que se desarrollen para IA-64 para que estos puedan aplicarlas a los códigos binarios IA-64 generados por los mismos. Dicho esto, queda claro que los programas de las plataformas Intel actuales no se beneficiarán de las cuatro unidades instaladas y que no obtendrán rendimientos espectaculares.

Como compensación a todo lo explicado, los diseñadores del micro darán soporte mediante hardware especial y soft de preprocesado, que ayudará a paralelizar y acelerar los binarios actuales, aunque claro está, en muchos tipos de algoritmos y demás, sólo se conseguirán resultados realmente buenos escribiendo fuentes nativos IA-64 con las optimizaciones pertinentes.

Se sabe ya que habrá como mínimo dos variantes derivadas de esta CPU

Para acabar de completar las ayudas, estos nuevos micros soportarán también la tecnología MMX con ampliaciones en lo que a cantidad de tipos de datos soportados se refiere y en la propia flexibilidad de manejo se refiere, lo cual dará mucha potencia a los algoritmos de tipo multimedia (por dar un término general).

El resultado de incorporar cuatro unidades de procesado para el set IA-64 , será que aparte de hacer menos secuencial la ejecución de las instrucciones de los programas (pues a partir de ahora tendremos que hablar de ejecución paralelizada), se acelerarán aquellos puntos de los programas donde más tiempo de reloj se consume, y que no es otra cosa que los bucles de iteración (*Loops*) de gran tamaño. Gracias a estas unidades de procesado y a las normas que aplicarán los nuevos compiladores a la hora de ordenar los binarios, se podrá hacer que en cada iteración del bucle de un algoritmo, se puedan procesar hasta dos o cuatro datos simultáneamente, lo cual da como resultado práctico una aceleración de x2 o x4 en la velocidad de procesado de bucles críticos, como pueden ser algoritmos multimedia y similares, donde los strings a

tener que manipular matemáticamente son muy comunes.

Lista de nuevos términos IA-64

A continuación, listamos algunos de los principales términos que aparecerán junto con esta nueva tecnología:

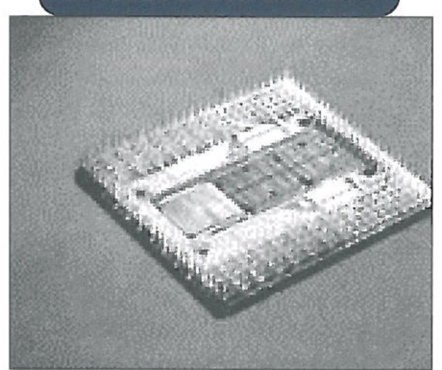
- 64 bits extension: Es el set de instrucciones de 64 bits, llamado comúnmente IA_64
- Extension Processor: Es el propio procesador IA-64.
- Extension Register: Es el nombre que se da a los registros de 64 bits usados por el set de instrucciones IA-64.
- Event: Como ya se ha mencionado anteriormente, a partir del Merced, el término *interrupción* queda substituido por el de *Evento*, lo cual es como actualizar el término a los tiempos actuales.
- XIP: Es el registro interno de 64 bits que hace las funciones de puntero extendido de instrucciones, que correspondería al EIP de la arquitectura x86 (recordad el famoso CS:EIP)
- XIP1: Un registro donde se salva el contenido del registro XIP cuando se genera un Evento (interrupción).
- XPCR: Registro extendido de banderas usado en IA-64 y que correspondería al registro EFLAGS en arquitectura x86.
- XPCR.ISA: El bit ISA del registro XPCR se usa para indicar el modo actual de funcionamiento del procesador.
- XPCR.d86i: Si este bit está a uno, significa que el procesador en cuestión no puede ejecutar instrucciones del set x86.
- XPCR.d86r: Si este bit está a 1 , el procesador no implementa registros x86 físicos.
- XPCR.dxi: Si este bit esta a 1, significa que el procesador no puede ejecutar instrucciones IA-64.
- XPCR1: Es un registro donde se almacena el contenido de XPCR cuando se produce un evento (una interrupción)

Nuevas instrucciones del IA-64

Como sabemos, este nuevo microprocesador incorporará un nuevo set de instrucciones además del propio set x86 clásico presente en los micros actuales. Seis de las nuevas instrucciones, que incorporará el chip, están destinadas a poder transferir información entre los registros x86 y los extendidos de "64 bits", tanto desde los primeros hacia los segundos como viceversa, aunque en el segundo caso, dichas transferencias de información sólo se pueden realizar desde las aplicaciones IA-64 y no desde programas x86 o sea, cuando la CPU esté funcionando en modo IA-64.

- **Instrucciones de conversión:** Como es de suponer, estas instrucciones podrán respetar el signo de los datos cuando la transferencia sea desde registros de 32 bits hacia registros de 64 bits, al igual que funcionan las instrucciones Intel x86 **MOVSX** y **MOVZX** cuando convierten datos entre registros x86 de 16 y 32 bits (para los que tengan curiosidad, estas instrucciones Intel se encuentran a partir de los chips 80386).

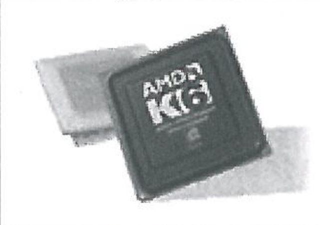
El Pentium Pro tampoco tiene los días contados. Acaba presentarse una versión con 1 MB de caché L2



El set de instrucciones MMX también estará soportado por Merced

- **Instrucciones de salto:** En este nuevo micro se dispondrá de dos instrucciones de salto para poder

No se sabe como reaccionará la competencia ante P7, pero de momento tan sólo intentan combatir los productos Intel que van apareciendo al mercado



The Superior Engine for Windows® Computing

AMD-K6™ MMX™ Enhanced Processor

cambiar la ejecución en curso de un programa x86 hacia código binario IA-64 y viceversa. La instrucción para pasar de código x86 a código IA-64 se llamará **JMPX86** y funcionará básicamente igual que el resto de salto del set x86, soportando incluso tanto los saltos directos como indirectos que se pueden realizar en las instrucciones clásicas x86. La instrucción de salto que podremos usar desde los binarios IA-64 para saltar hacia código x86 se llama **x86JMP**. Para que los lectores más expertos se hagan una idea más clara de como funcionará este micro por dentro, a continuación detallaremos un poco su funcionamiento, en el caso de estas dos instrucciones.

Instrucción x86JMP

Cuando se ejecuta esta instrucción, se pone la CPU en modo x86, y se leerá entonces la siguiente instrucción que se encuentre a partir de la dirección que se le haya indicado. La fórmula que realiza para convertir la dirección del puntero de instrucción, es **XIP = XIP+ (rel17*4)-CS_BASE**, donde se ve que en **(rel17*4)** se expande el signo y se multiplica el resultado por cuatro. El **XIP** resultante es un puntero truncado a 32 bits.

Las primeras versiones del micro funcionaran como mínimo a 250 Mhz

Por ejemplo, si tenemos el caso de que poseemos un segmento de código iscr[47:32] y un offset de 32 bits iscr1[31:0], Si EFLAGS.VM86 está a 1, el procesador pondrá el selector de 16 bits cuatro bits hacia la izquierda. Si el EFLAGS.VM86 valiese cero, entonces el procesador leería el descriptor de CS de la LDT/GDT y comprobaría si se ha producido un caso de violación de segmentación.

Instrucción JMPx86

El funcionamiento interno de la instrucción de salto desde x86 a IA-64 es bastante más sencillo. El único problema que representa técnicamente para los diseñadores es que tiene que poder tener dos modos de salto, el llamado salto directo y el relativo.

- Salto directo: cuando se realiza un salto de ejecución relativo, la CPU calcula la dirección de 64 bits a partir del contenido actual del registro XIP y del código base. Por ejemplo: **XIP = XIP = rel32/16 = CS_Base**, quedando patente que la dirección resultante es totalmente lineal, una dirección física real de 64 bits. Después sólo queda truncar XIP a 32 bits.
- Salto relativo: cuando el salto se realiza con el método directo, se lee un registro o una dirección de memoria especificada por r/m32/16 y se busca la dirección lineal de dicho valor. Entonces sólo queda truncar XIP a 32 bits.

Por otra parte, JMPX no modifica el nivel de privilegio en que se encontrase el procesador y espera a que todas las operaciones de coma flotante x86 se hayan concluido para evitar que se produzcan errores de cálculos.

En caso de que XPCR.dxi valiera 1, la instrucción se anula y se genera un error. En el caso de que las extensiones de 64 bits del microprocesador estuviesen inactivas, la instrucción produciría una ejecución ilegal.

Interrupciones

Aun que parezca extraño a primera vista, en este nuevo micro siempre que se produce una interrupción se pone el microprocesador en modo de funcionamiento x86 y se ejecuta obligatoriamente por lo tanto, una rutina con binarios (instrucciones) x86.

Dicha esta *directiva* de funcionamiento, tenemos como consecuencia de este diseño que todos los punteros de interrupción (vectores) deben contener una instrucción de salto del tipo **JMPX.**

Además de este detalle, como nos encontramos que dichas interrupciones (que ahora tendremos que llamar *eventos* como se explicará más adelante) han podido ser generadas tanto cuando se estaba ejecutando un programa x86 o un IA-64, la instrucción de retorno x86 que se incluye al final de todas las rutinas de interrupción deberá poseer dos versiones distintas de microcódigo para poder retornar a uno u otro tipo de código según sea el caso.

El rendimiento de este micro será varias veces superior a los actuales

La instrucción de retorno, continua llamándose como siempre **IRET** y su lógica interna es la misma que la versión antigua, con el añadido de que examina la bandera llamada **XPCR.ISA** (flag). Esta bandera o flag, registra si la interrupción se produjo en modo x86 o IA-64. En caso de que dicha bandera valga 0, IRET funciona de forma normal (x86) como le corresponde en ese modo y en caso de ser 1, salva las banderas en el registro XPCR y carga el contador de programas en el registro **XIP**.

Tamaño de los registros

Todos los registros enteros usados por el set IA-64, son como hemos dicho ya, de 64 bits al igual que el contador del programa (puntero de instrucciones), aunque para los tipos flotantes se sigue usando el tipo usado en los Intel actuales, los cuales recordemos, poseen una longitud interna de 80 bits por registro.

Para los más curiosos desconocedores de este tema, recordar sólo que el sistema de registros de 80 bits de la FPU (Floating Point Unit) usa de izquierda a derecha: 1 bit para signo, 15 bits para el exponente, 1 bit para indicar si es un normalizado o no y los 64 bits restantes para la mantisa.

Tabla 2. Resumen de las características

Como resumen de todo, se van a resumir todas las principales características de este micro:

- Poseerá arquitectura WLSI, y habrá 3 versiones iniciales del mismo, el primero de los cuales se llama Merced, el segundo no se sabe todavía y el tercero Tahoe.
- El primero de ellos lo venderá Intel y será el primero en aparecer.
- El segundo poseerá un set de instrucciones incorporado de PA-RISC para poder ejecutar el sistema operativo UNIX-HP
- El tercero, que posee el nombre temporal de *Tahoe,* será el que se usará según anuncian sus desarrolladores como el chip de la plataforma base que se usará para el futuro sistema operativo Windows de 64 bits.
- Este microprocesador , estará formado por unos ocho millones de transistores, cifra sensiblemente superior a los 3.3 millones del Pentium clásico y los 5.5 del Pentium II.
- Disipará unos 40 Watts, aunque éste es un dato orientativo.
- Debido a su mayor simplificación de diseño interno, podrá funcionar a mayor velocidad y se dice que los modelos iniciales funcionarán como mínimo a velocidades de reloj de 250 Mhz.
- Este chip también se caracterizará por poseer un menor consumo de energía y como ya sabemos, será compatible con todo el software actual tanto de 16 como de 32 bits.

Funcionamiento de SET X86

Cómo estará diseñado el microprocesador internamente aun no se sabe demasiado bien y sólo se pueden realizar hipótesis a partir de las diversas patentes que ha registrado Intel sobre diversas tecnologías de posible implementación en dicho micro.

En Merced, todas las rutinas de interrupción deberán ser en código x86

Una de las posibilidades barajadas para el funcionamiento de la parte x86 de la CPU, es que el microprocesador poseerá un sistema interno de traslado x86 a IA-64 (conversor), que las convertirá a código IA-64 antes de ser ejecutadas y cacheadas (almacenadas en la memoria caché). Otra posibilidad podría ser que el micro poseyera una sola unidad de procesado integrada y que ambos sets de instrucciones se ejecuten en dicha unidad (lo cual es muy improbable).

En lo que a registros internos se refiere, hay dos posibilidades claras a poder implementar en este microprocesador: la primera posibilidad es que se creen dos sets de registros físicamente independientes para cada set instrucciones del procesador. La otra posibilidad es que los registros x86 se diseñen como alias que se definirían dentro de los registros IA-64 (la combinación inversa sería imposible físicamente por tamaño de los elementos).

Hay que recordar, que un caso de diseño similar al mencionado basado en Alias ya existe en la actualidad y que lo encontramos en la tecnología MMX implementada en los Pentium MMX y Pentium II actuales. Para el que no conozca como funcionan, puede remitirse al artículo sobre MMX, que se publicó en la revista número 33. Recordar que los registros de 64 bits, para usar con las instrucciones MMX, eran en realidad físicamente los mismos que los que usaba la unidad de coma flotante para usar como pila de registros FPU, es decir, que los registros MMX son alias de los registros de la FPU (los llamados *ST(x)*).

Fecha de aparición

Intel aún no ha dado ninguna fecha orientativa de cuando saldrá el nuevo microprocesador al mercado, y pese a que no se compromete a nada hasta el año 2000, ya hay rumores de que podría salir a finales de 1998 o durante 1999.

Aun no se sabe si los registros x86 serán alias de los IA-64 o independientes

Por otra parte, Intel ha querido dejar también claro, que la causa de que no quiera dar una fecha de presentación del producto, no significa que esté teniendo problemas en su desarrollo (retrasos inesperados), si no que es puramente una problemática de estrategia de mercado.

Como suele ser habitual cuando aparece un nuevo microprocesador, los primeros sistemas que lo incorporarán serán estaciones de trabajo y servidores. Así

por ejemplo, tenemos que Hitachi será una de las primeras marcas que presentará un servidor basado en este chip en cuanto este disponible, destinado a usarse tanto con UNIX como con Windows NT. Hay otras compañías que han anunciado su interés en incorporarlo en sus productos, como son Hewlett Packard, Compaq, Siemens Nixford y Net power entre otras.

Otros datos de interés

Según se ha confirmado, Intel será la única compañía que venderá un modelo de dicho microprocesador, que se llamará explícitamente Merced , ya que como se ha dicho ya, habrá diversas variantes de dicho microprocesador. Por otro lado, Hewlett Packard lanzará un versión derivada del mismo que incluirá un set de instrucciones PA-RISC y que estará optimizado especialmente para ejecutar el sistema operativo UNIX y según ha dicho la propia compañía, las aplicaciones actuales de 32 bits que operan con HP-VX o SCO Openserver podrán ser ejecutadas en este nuevo microprocesador pasando por una simple previa recompilación.

El futuro Windows 64 funcionara sobre la variante del chip llamada Tahoe

La última variante conocida de momento será una versión que están desarrollando también conjuntamente Intel y Hewlett Packard bajo el nombre temporal de *Tahoe* y que será el chip que se incorporará en las futuras plataformas de 64 bits que usarán el futuro Windows de 64 bits.

Otros microprocesadores

Aunque normalmente la guerra de microprocesadores entre Intel, AMD y CYRIX, siempre ha sido muy dura, con el diseño actual de este nuevo "microprocesador-doble" de nueva tecnología, la competencia parece que ha quedado momentáneamente un poco a la espera de lo que pasará en el futuro y, de momento, simplemente se limitan a ir intentando presentar modelos de microprocesadores que intenten competir en prestaciones y compatibilidad con los últimos modelos que Intel ha sacado al mercado.

Se dice que los primeros modelos funcionarán como mínimo a 250 Mhz

Por ejemplo, tenemos que el último producto que ha desvelado AMD, el K6, está destinado a competir, como sus propios diseñadores han comentado, con el Pentium II de Intel, al igual que pasa con el nuevo microprocesador M2 de la casa Cyrix, que está también diseñado para intentar arrebatar mercado a Intel en el terreno del Pentium II.

Conclusiones

Aunque la decisión tomada por Intel y HP, de crear un microprocesador con doble set de instrucciones para realizar una renovacion tecnológica no traumática, ha sido la mejor solución que se podía tomar, hay algunos aspectos que quedan un poco borrosos en la parte técnica.

Como ya se ha dicho, las interrupciones siempre tienen que ejecutarse en modo x86, puesto que ello hace que las aplicaciones actuales puedan funcionar en la nueva plataforma. Pero este diseño, seguramente creará un problema cuando se vayan creando cada vez más aplicaciones puras de IA-64 bits y se vaya abandonando la programación de binarios x86. A medida que transcurra el tiempo, este diseño se convertirá en un obstáculo, ya que obligará a los programadores a tener que seguir creando código x86 para la gestión de interrupciones de sus propias aplicaciones, lo cual, como sabemos , hará que baje el rendimiento del equipo globalmente, puesto que estos nuevos microprocesadores estarán optimizados y diseñados básicamente para sacar todo el provecho en modo IA-64 y el tener que cambiar de modo de procesador cada vez conllevará asociados muchos ciclos de reloj perdidos.

Más información

Si se desea obtener más información sobre este microprocesador, se puede acceder a multitud de páginas Web donde se encontrarán numerosos artículos sobre el propio chip y temas relacionados, aunque para empezar, la propia Web de Intel es el lugar más recomendado **<http://www.intel.com>**. Otro lugar recomendado está en la dirección http://chipanalyst.com/mpr/merced/index .html y en **< http: // chipanalyst.com/ mpr/ merced/ 1017vp.html >** donde el lector encontrará un artículo bastante interesante sobre este microprocesador.

Si el lector desea encontrar más información sobre los nuevos procesadores de Intel, puede acceder a los buscadores de la red e introducir términos como Merced o IA-64. Un buscador donde el lector puede encontrar bastantes enlaces o links, es el HotBox (www.hotbox.com) y los siempre recomendados OLE y OZU.

Si el lector quiere contactar con el autor o buscar más información , puede hacerlo en la WEB de programadores **<www.arrakis.es/~erde>** o enviando un e-mail a la dirección **<erde@arrakis.es>**

¿TE ATREVES?

Desde **Sólo Programadores** estamos preparándonos para un torneo muy especial, de ordenador a ordenador, de programador a programador. El torneo de ajedrez está empezando a tomar forma y éstas son las bases, aunque también podéis informaros en AWS en la calle Serrano Jover, 3, de Madrid, o en el teléfono (91) 5 42 50 87. Para conocer cuál será el alcance de la participación, os pedimos que os inscribáis enviando vuestros datos a la dirección de AWS o a la dirección de e-mail: **solop@towercom.es.** El periodo de inscripción ya ha comenzado y terminará el 30 de noviembre, y la fecha límite para la recepción de los programas es el 15 de diciembre de 1997. La celebración del torneo será anunciada con suficiente antelación y podréis participar en persona o desde vuestra propia ciudad, ya que vuestro programa os representará y en todo momento estará presente un notario.

BASES DEL TORNEO

1 Se presentará un programa que juegue al ajedrez recibiendo la jugada del contrario y devolviendo la propia. El tiempo de la partida será medido por un juez y la partida se jugará físicamente en una mesa con un tablero siguiendo las instrucciones que cada ordenador dé; cada ordenador avisará de que ha elegido un movimiento con un pitido y mostrará su jugada en la pantalla de la forma: casilla a mover - casilla destino. La notación de la casilla vendrá dada por su columna y su fila. Para las columnas se utilizarán las de la A a la H siendo la columna A la primera columna de la izquierda, visto el tablero desde la posición de las blancas. Para las filas utilizaremos los números del 1 al 8 siendo la fila 1 la más cercana a nosotros (visto el tablero desde el lado de las blancas). Como ejemplo, el movimiento del peón de rey dos lugares hacia adelante sería: "E2 - E4". La jugada que hace el contrario se anotará en el ordenador de forma manual tecleando "E2-E4" <RETURN>.

2. El programa jugará al ajedrez según las normas de este juego, admitiendo todos sus movimientos.
3. El programa se ejecutará bajo Windows 95, MS Dos 6.22 y Linux, en un Pentium 166 con 16 MB de RAM.
4. El programa podrá jugar con las blancas o con las negras indistintamente.
5. Se debe entregar el código fuente, la versión del compilador utilizado y una breve documentación para su utilización.
6. Un programa podrá ser descalificado si realiza un movimiento no válido y esto deberá ser advertido por el contrario; por lo tanto, un movimiento será válido siempre que el otro programa no nos advierta de lo contrario. De la misma forma, si un programa nos advierte de un movimiento erróneo y éste no lo es, será descalificado.
7. Los jueces son los encargados de advertir las posiciones de tablas, el programa no advertirá de ello.
8. El programa debe detectar si ha ganado la partida y quedará descalificado ante cualquier detección errónea. No se avisará al contrario en el caso de que deje al rey al descubierto y ganará la partida el que antes mate al rey.
9. La metodología del concurso se reservará hasta que sea conocido el número de participantes (eliminatoria, puntos, etc.)
10. No se podrán utilizar en el programa bases de datos adicionales.
11. Los derechos del programa ganador pasarán a la propiedad de la empresa Tower Communications, SRL.
12. El premio consistirá en un Pentium PRO 200 con 64 MB RAM, VIRGE4 MB, CDX 16, disco duro 5 GB, ~~monitor 17", teclado, ratón~~ y disquetera.

ANEXO A LAS BASES

1. La velocidad de juego se establece en 2 horas (1 hora por jugador).
2. Las jugadas que el árbitro considere erróneas eliminarán al programa que las haya indicado.
3. El programa debe ser original.
4. La fecha de recepción del fuente y el compilador se establece entre el 1 y el 31 de Enero. Se deberán remitir a la siguiente dirección: AWS Informática, Serrano Jover 3, 28.015, Madrid, a la atención de Fernando Núñez.
5. El primer día de celebración del torneo será el 8 de Febrero de 1998 en la dirección indicada en el punto anterior.
6. Debido a las peticiones recibidas por los participantes, se admiten bases de datos adicionales.

CONCURSO DE PROGRAMACIÓN

SÓLO PROGRAMADORES Y AWS INFORMATICA

SERRANO JOVER 3, MADRID

Inteligencia Artificial: Simbólico versus Subsimbólico

Manuel de la Herrán Gascón
mherran@usa.net

Aunque no es algo habitual en nuestra revista, en este número hemos incluido un artículo de opinión en el que además se da un repaso a las distintas tendencias en Inteligencia Artificial, introduciendo al lector en el estado del arte de esta nueva ciencia.

"¡Tachaaan! Señoras y señores: he aquí la lavadora inteligente, ..." ¿Cómo? ¿Lavadora? ¿Alguien ha dicho Java? Ya sabemos que Java era originariamente un lenguaje para electrodomésticos, pero esta vez no se trata de eso. Esta columna trata de otra cosa. El título podría haber sido "La Inteligencia Artificial (IA) ha muerto" o "Lo de la IA ya no se lo cree nadie".

Y es que nos hemos pasado. Tenía que ocurrir. Ahora, si alguien califica su creación informática de inteligente, se fruncen ceños de desconfianza, cuando no se provoca risa. Rodeados como estamos de edificios inteligentes que nos obligan a llevar jersey en verano y pantalón corto en invierno, automóviles inteligentes que insisten en que nos pongamos un cinturón de seguridad estropeado, y **semáforos inteligentes** en perenne madurez y sin caer del guindo, ha llegado un momento en el que decir que nuestro programa es inteligente sólo lo descalifica.

Lo de inteligente ya no se lo cree nadie

La culpa la tiene, por supuesto, la televisión, la publicidad y las lavadoras automáticas. Tras el éxito indiscutible del iterado cliché de la vecina elogiando **las camisas de su marido**, los comerciales informáticos no dudaron en aplicar similares atribuciones a sus productos. Los jabones pasaron del superblanco al ultrablanco. Nuestros programas, con idéntica lógica, de ser *Sistemas Expertos* o *Agentes Inteligentes*, a usar tecnología *IntelliSense* (un tipo de ayuda sensible en el que trabaja Microsoft). También han salido ya las lavadoras con *Fuzzy Logic*, concretamente la nueva Eco-Lavamat de AEG. No dudo que sea una fantástica lavadora. Probablemente la capacidad de manejar conceptos difusos mediante *Lógica Borrosa* le confiera alguna ventaja; tal vez la de distinguir la ropa sucia de la muy sucia y de la extremadamente sucia. Pocas cosas hay tan borrosas como una camiseta blanca después de diez días de travesía por el monte, sin mas agua que la de la cantimplora. Poco importa, en definitiva, el nombre del jabón, se trata de que lave.

¿Y ya lava? No está mal. ¿Y es inteligente? Pues vaya, depende... Pero bueno, ¿Es que cada uno tiene derecho a llamar IA a lo que le dé la gana? Pues claro que sí. Esto ocurre porque nadie sabe aún qué demonios es la IA. Yo quiero mostrar en primer lugar mi definición predilecta, mediante una inspirada conversación de bar, real como la vida misma.

– ...y acabo de terminar un programa inteligente. -Comento bajito, para que no me oiga nadie más, claro.

– ¿Te refieres a un programa de IA? -Me responde- ¡Eso es imposible!

Ésta es mi oportunidad de alargar un poco más la noche, y llenar de nuevo el vaso de cerveza vacío. Sin vacilar comienzo la habitual argumentación.

- Bueno, hay gente que opina que nunca se podrá imitar la mente humana, pero...

- ¡No, No! -Me corta enseguida- Todo eso me lo creo. ¡Lo que digo que es imposible es que lo hayas hecho tú!

¡Acabáramos! La IA son los últimos descubrimientos, las nuevas tecnologías, los límites de la computación. La IA es una frontera en continuo movimiento. Y claro está, parece obvio que todo esto no es posible en **manchego**: si no lo leo en inglés, no me lo creo.

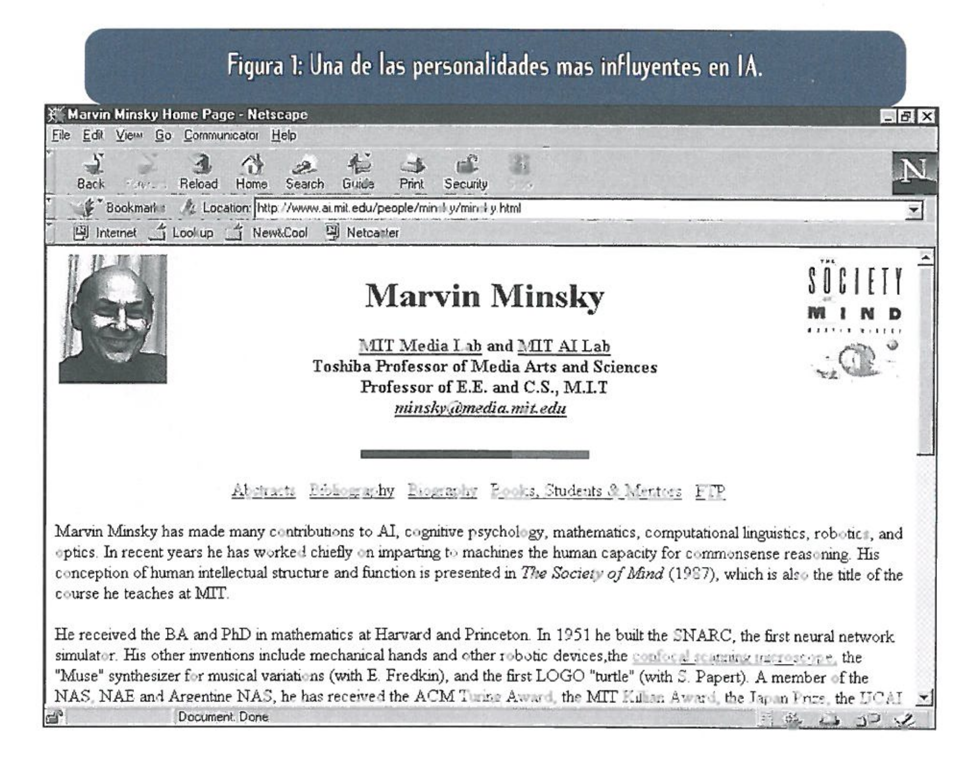
Hay otras definiciones más aceptadas: la IA trata de construir máquinas con comportamiento aparentemente inteligente. El hombre es un ser inteligente. ¿Lo son los animales? Podemos decir que en torno a la respuesta a esta pregunta surgen los dos grandes bloque enfrentados en la materia: el enfoque simbólico o Top-Down, también llamado *IA clásica*, y el enfoque subsimbólico (Bottom-Up) o conexionista.

Los *simbólicos* simulan directamente las características inteligentes que se pretenden conseguir. Como modelo de mecanismo inteligente a imitar, lo mejor que tenemos y más a mano es el Hombre (que tal vez no sea gran cosa, todo hay que decirlo). Desde este punto de vista, poco interesa simular los razonamientos de los animales. El *boom* de los Sistemas Expertos, ahora de capa caída, fue producido por este planteamiento.

Las principales corrientes en IA son la simbólica y la subsimbólica

Para los constructores de sistemas expertos, es fundamental la representación del conocimiento humano y debemos a ellos los grandes avances en este campo.

Figura 1: Una de las personalidades mas influyentes en IA.



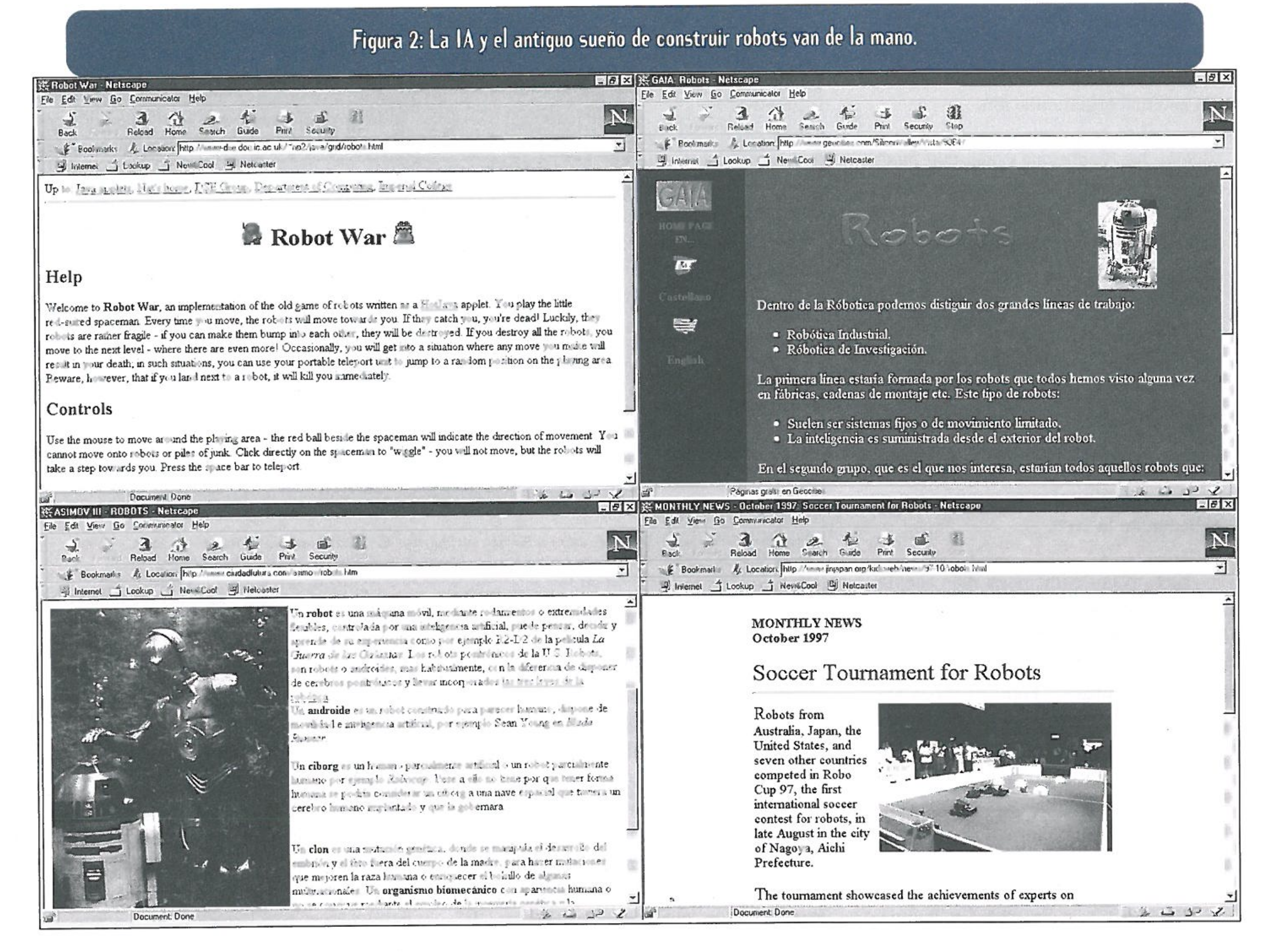
Esto es debido a que, realizando una gran simplificación, se debe incluir en un sistema experto dos tipos de conocimiento: "conocimiento acerca del problema particular" y "conocimiento acerca de cómo obtener más conocimiento a partir del que ya tenemos". Para el primero existen técnicas como los Frames (marcos) que fueron los padres de lo que hoy conocemos como *Programación Orientada a Objetos*. El segundo es llamado también *mecanismo de inferencia* y requiere además de un *método de búsqueda* que permita tomar decisiones, como por ejemplo, seleccionar la regla a aplicar del conjunto total de posibles reglas. Esto puede parecer lo más sencillo, pero puede ser lo más difícil: se trata de elegir y elegir bien, pero sin demorarse varios millones de años en hacerlo.

Como ejemplo representativo de la rama simbólica llevada al extremo tenemos el proyecto Cyc de Douglas B. Lenat, con un sistema que posee en su memoria millones de hechos interconectados. Según Lenat, la inteligencia depende del numero de reglas que posee el sistema, y *"casi toda la potencia de las arquitecturas inteligentes integradas provendrá del contenido, no de la arquitectura"*. Para él, los investigadores que esperan poder resolver con una única y elegante teoría todos los problemas de inferencia y representación de conocimientos, padecen celos de la física: ansían una teoría que sea pequeña, elegante, potente y correcta.

Los esfuerzos de la otra rama de la IA, los subsimbólicos, se orientan a simular los elementos de mas bajo nivel que componen o intervienen en los procesos inteligentes, con la esperanza en que de su combinación emerja de forma espontánea el comportamiento inteligente. Los ejemplos más significativos probablemente sean las *Redes Neuronales Artificiales* y los *Algoritmos Genéticos*. Aunque parezcan un fenómeno reciente, estos paradigmas no son más jóvenes que los *Sistema Expertos* de la IA clásica, simplemente tuvieron menor publicidad y financiación. El Primer modelo de red neuronal fue propuesto en 1943 por McCulloch y Pitts. El Perceptrón de Rosenblat apareció en 1959, produciendo una gran y breve expectación que quedó pronto en el olvido, y J. H. Holland introdujo la idea de los *Algoritmos Genéticos* en los años sesenta. Las grandes ventajas de estos sistemas son la autonomía, el aprendizaje y la adaptación, conceptos todos ellos relacionados.

Las peleas y críticas entre los dos planteamientos han sido casi tan intensas

Figura 2: La IA y el antiguo sueño de construir robots van de la mano.



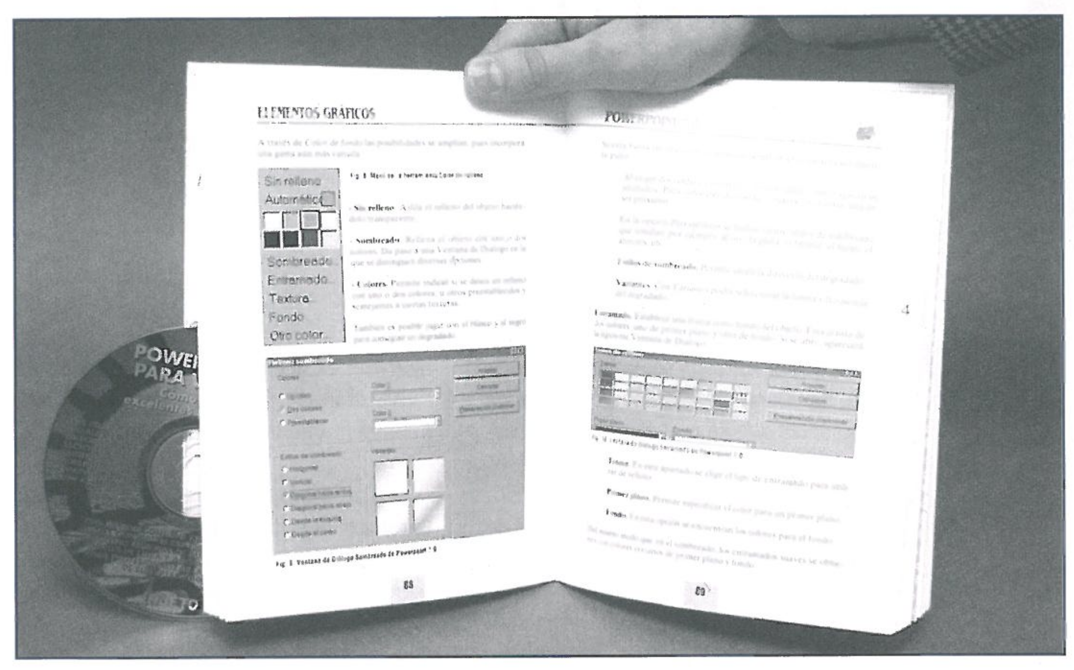
como ridículas. Tras la resaca del auge de la IA simbólica, todos los éxitos han sido cosechados por los subsimbólicos, y el "pasarse al otro bando", al menos en ciertos aspectos, está siendo una actitud habitual. Como era de esperar, los científicos pueden ser testarudos, pero no son tontos, y finalmente triunfa el sentido común de aprovechar las grandes cualidades de una y otra postura, en sistemas combinados.

Referencias

- *Curso de Doctorado en Inteligencia Artificial*. Isasi, Pedro. Universidad Carlos III.
- *"Sistemas expertos en contabilidad y administración de empresas"*. Sierra Molina, Guillermo, y otros. 1995. Editorial Ra-Ma.
- *" Bebés de silicio "*. Wallich, Paul. Revista Investigación y ciencia, Febrero 1992.
- *"Estrategias Evolutivas"* en http://gaia.uc3m.es/~elmaor/estrategias_evolutivas.html
- *"Conceptos básicos sobre RNA"* en http://www.gc.ssr.upm.es/inves/neural/ann2/concepts/concepts.htm
- *"Genetic Algorithms In The World"* en http://www.lcc.uma.es/personal/alba/services/nag2.html
- Seminario sobre Computación Evolutiva en http://www.lcc.uma.es/personal/cotta/wceta97/wceta97.html
- Appendix to Ray Kurzweil's book *"Intelligent Machines"* (MIT Press, 1990) (A timeline of the history of AI).
- Pamela McCorduck, *"Machines Who Think"*, Freeman, San Francisco, CA, 1979.
- Allen Newell, *"Intellectual Issues in the History of Artificial Intelligence"*, Technical Report, Carnegie Mellon University Computer Science Department, 1982.
- Charniak and McDermott's, *"Introduction to Artificial Intelligence"*, Addison-Wesley, 1985
- Daniel Crevier, *"AI: The Tumultuous History of the Search for Artificial Intelligence"*, Basic Books, New York, 1993.
- Henry C. Mishkoff, *"Understanding Artificial Intelligence"*, Howard W. Sams & Co., Indianapolis, 1985 .
- Margaret A. Boden, *"Artificial Intelligence and Natural Man"*, Basic Books, New York, 1987.

La mayoría de anuncios editoriales te muestran los libros cerrados

NOSOTROS HEMOS PENSADO QUE PREFIERES VERLOS ABIERTOS



Libro + CD-ROM
por sólo
1.995
ptas.
IVA incluido

- ✓ Explicaciones paso a paso
- ✓ Ejemplos prácticos a todo color
- ✓ Imágenes de cada una de las pantallas que encontrarás en tu ordenador



Títulos que componen la colección "Biblioteca Informática Avanzada":

- EXPLORANDO INTERNET
- 3 D STUDIO V.4
- AUTOCAD 13
- MÚSICA DIGITAL POR ORDENADOR
- REALIDAD VIRTUAL
- LENGUAJES HTML, JAVA Y CGI
- WORD PARA WINDOWS 95
- COREL 6.0
- VISUAL BASIC 4.0
- POWERPOINT 7.0 PARA WINDOWS 95
- EXCEL 7.0 PARA WINDOWS 95
- JAVA. EL LENGUAJE DE INTERNET
- MICROSOFT WORD 97
- MICROSOFT ACCESS 97
- MICROSOFT EXCEL 97

Distribuidores autorizados en librerías

COMUNIDAD DE MADRID / CASTILLA LA MANCHA	
DISTRIFORMA S.A.	Tfno. 91 - 501 4749
CATALUÑA	
MIDAC LLIBRES S.L.	Tf. 93-421 18 95
ANDALUCIA OCCIDENTAL/ EXTREMADURA	
DISTRIBUCIÓN DE EDICIONES RDGUEZ. SANTOS S.L.	Tfno. 95 - 418 04 75
ANDALUCIA ORIENTAL	
DISTRIBUCIONES DEL MEDIODIA S.A. (ZÓCALO)	Tfno. 958-550278
CASTILLA - LEÓN	
ARCADIA S.L.	Tfno. 983-395049
GALICIA	
LUIS REY ABELLA (DISGALIBRO)	Tfno. 981-795754
ASTURIAS - CANTABRIA	
DISTRIBUCIONES CIMADEVILLA S.A.	Tfno. 98-5167930
CANARIAS	
GARCIA PRIETO LIBROS S.L.	Tfno. 922-820026
BALEARES	
PONENT LLIBRES S.L.	Tfno. 971-430339
VALENCIA - CASTELLÓN	
ADONAY S.L.	Tfno. 96-3793151
ALICANTE - MURCIA - ALBACETE	
LA TIERRA LIBROS S.L.	Tfno. 96-5110192
PAÍS VASCO - NAVARRA	
YOAR S.L.	Tfno. 948-302239
ARAGÓN - LA RIOJA	
ICARO DISTRIBUIDORA S.L.	Tfno. 976-126333

ABETO c/ Aragoneses, 7. 28.108 Alcobendas (MADRID)
Tel.: (91) 661 42 11* Fax: (91) 661 43 86

Curso de Programación en Perl (II)

Ernesto Schmitz
Chema Álvarez

Con este artículo vamos a dar fin al curso que iniciamos el pasado mes en el especial Internet, sobre programación en Perl, un lenguaje que sonará bastante a todo buen usuario de Internet y más todavía a los que están haciendo sus pinitos en el campo de la creación de contenidos Internet-intranets, ya sean páginas HTML, formularios, etc...

En la anterior entrega sentamos las bases sobre los tipos de datos, las funciones más usuales que se realizan con las variables, sus operadores y mostramos varios ejemplos de la utilización de instrucciones condicionales y manejo de ficheros de texto (entrada/salida).

Veremos a continuación sustituciones y traducciones, arrays asociativos, subrutinas o funciones y gestión de cadenas de texto.

Y es que una de las características más útiles de Perl (sino la más útil) es su potente capacidad de manipulación de cadenas. En el corazón de esto, se encuentra la expresión regular (RE), que es compartida por otras muchas utilidades Unix.

Expresiones Regulares

Por ejemplo, si queremos averiguar si una cadena está contenida en otra, utilizaremos expresiones regulares que indicarán a Perl la cadena de búsqueda.

```
$frase =~ /hacer/
```

Con esta simple operación estamos comprobando si la cadena mostrada entre barras de división está incluida en la cadena frase; es decir, si la cadena frase fuera "Viva el Perl", la sentencia anterior nos hubiera devuelto el valor lógico falso.

Por supuesto, como en todos los entornos heredados de Unix, las expresiones regulares (RE) son *case sensitive* (sensibles a mayúsculas-minúsculas).

```
$frase = "Hacer algo con Perl";

$frase =~ /hacer/
```

Esta última expresión retornaría falso como valor booleano.

Existe otro operador usado para localizar las no coincidencias. Por ejemplo, en el caso anterior, si modificásemos la última sentencia por

```
$frase !~ /hacer/
```

obtendríamos una respuesta afirmativa, puesto que "hacer" no está contenido dentro de frase.

La variable especial $_

Podríamos usar un condicional de la siguiente manera

```
if ($frase =~ /cocer/)
```

```
{
        print"No sabes ni freir un huevo\n";
}
```

que mostraría el mensaje en cualquiera de los siguientes casos

```
$frase= "cocer se me da bien";
$frase= "cocer y freir no es lo mismo";
```

pero a menudo es mucho más fácil si asignamos la frase a la variable especial **$_**, que es por supuesto un escalar. Si hacemos esto no necesitaríamos usar los operadores de coincidir y no coincidir con lo que el anterior condicional podría haber sido escrito así

```
$_ = "cocer se me da bien";
if (/cocer/)
{
        print"No sabes ni freir un huevo\n";
}
```

la variable **$_** es usada por defecto en un montón de operaciones en Perl y se tiende a utilizar en muchos casos.

Los corchetes son usados para hacer coincidir cualquiera de los caracteres dentro de ellos

En una expresión regular podemos usar un montón de caracteres, esto es lo que hace de ellas unas herramientas muy potentes, pero al mismo tiempo parecen difíciles de manejar.

Vamos a entrar lentamente en materia, la creación de expresiones regulares utilizando estos caracteres puede ser todo un arte. En la Tabla-1 incluimos unos cuantos de estos caracteres especiales y su significado.

A continuación tenemos un ejemplo de su uso, recordando, cómo no, que debemos encapsularlos entre barras de división.

```
t.e     # 't' seguido de cualquier cosa seguida de 'e'
        # Aquí encajarán "the", "tre", "tle" pero
        # no "te", "tale"
^f      # 'f' al principio de una línea
^ftp    # 'ftp' al principio de línea
e$      # 'e' al final de una línea
tle$    # 'tle' al final de una línea
und*    # 'un' seguido de cero o más caracteres
        # 'd'
        # Aquí encajarán "un", "und", "undd",
        # "unddd", etc...
.*      # Cualquier cadena sin nueva línea. Esto
        # es porque '.' coincide con cualquier
        # cosa excepto una nueva línea y *
        # significa cero o más de esto.
```

Existen más opciones. Los corchetes son usados para hacer coincidir cualquiera de los caracteres dentro de ellos. Dentro de los corchetes, un guión '-' indica "entre" y '^' al principio significa "not".

```
[qjk]       # ó 'q' ó 'j' ó 'k'
[^qjk]      # ni 'q' ni 'j' ni 'k'
[a-z]       # cualquier cosa entre la 'a' y la 'z',
              ambas
            # inclusive
[^a-z]      # ninguna letra en minúsculas
[a-zA-Z]    # cualquier letra
[a-z]+      # cualquier secuencia no vacía de letras
            # minúsculas
```

Una barra vertical "|" representa el OR y los paréntesis pueden ser utilizados para agrupar cosas:

```
cocer|freir    # ó "cocer" ó "freir"
(mej|pe)or     # ó "mejor" ó "peor"
(bla)+   # ó "bla" ó "blabla" ó "blablabla" ó...
```

aquí tenemos otros caracteres especiales más:

```
\n     # retorno de carro.
\t     # el tabulador.
\w     # cualquier carácter alfanumérico (palabra).
       # lo mismo que [a-zA-Z0-9_]
\W     # cualquier carácter no alfanumérico.
       # lo mismo que [^a-zA-Z0-9_]
\d     # cualquier dígito. Lo mismo que
       # [0-9]
\D     # cualquier no-dígito. Lo mismo que
       # [^0-9]
\s     # cualquier carácter invisible:
       # espacio, tabulador, nueva línea,
       # etc.
\S     # cualquier carácter no-invisible.
```

Tabla 1
De caracteres especiales.

```
.     # Cualquier carácter único excepto
        nueva línea
^     # Principio de línea o cadena
$     # Final de línea o cadena
*     # Cero o más del último carácter
+     # Uno o más del último carácter
?     # Cero o uno del último carácter
```

Claramente caracteres como '$', '|', '[', ')', '\', '/' y demás son casos peculiares en las expresiones regulares. Si se desea hacer coincidir alguno de ellos, entonces habrá que anteponerles el símbolo '\'.

Algunos ejemplos de expresiones regulares (RE)

La mejor manera de aprender a usar las expresiones regulares es cuidando enormemente su sintaxis, como ya se mencionó anteriormente. Recomendamos pues probar todos los casos antes de admitir una expresión regular como válida para un objetivo concreto. Seguidamente incluimos unos cuantos ejemplos. Recordamos de nuevo que el uso de ellas ha de ser encapsulado entre barras de división /.../

```
[01]      # "0" ó "1".
\/0       # Una división por cero: "/0"
\/ 0      # Una división por cero con un espacio:
          # "/ 0"
\/\s0     # Una división por cero con un carácter
          # invisible: "/ 0" donde el espacio puede
          # ser un tabulador...etc.
\/ *0     # Una división por cero con posiblemente
          # algunos espacios: "/0" ó "/ 0" ó "/  0",
          # etc...
\/\s*0    # Una división por cero con posiblemente
          # algún carácter invisible.
```

```
\/\s*0\.0* # Como la anterior, pero con un punto
           # decimal y algunos ceros (0s)
           # después. Se acepta "/0.", "/0.0",
           # "/0.00", etc... y "/ 0.", "/  0.0"
           # y "/   0.00".
```

Sustitución y Traducción

Al igual que cuando identifica expresiones regulares, Perl puede hacer sustituciones basadas en las propias coincidencias encontradas. La manera de hacer esto es usar la función **s**, la cual ha sido diseñada para emular la forma en las que estas sustituciones son hechas en el editor de texto "vi" de Unix.

Para sustituir la aparición de "madrid" por "Madrid" en la cadena **$frase**, usaremos la expresión

```
$frase =~ s/madrid/Madrid/
```

y para hacer lo mismo con la variable especial **$_**

```
s/madrid/Madrid/
```

Nótese que las dos expresiones regulares (madrid y Madrid) están rodeadas por un total de tres barras de división. El resultado de esta expresión es el número de sustituciones hechas, así que será 0 (falso) ó 1 (verdadero) en este caso.

Este ejemplo sólo sustituye la primera ocurrencia encontrada en la cadena, pudiendo existir más de una que queramos reemplazar. Para realizar pues una sustitución global tendremos que añadir al final de la sentencia la letra **'g'** de "global" así

```
s/madrid/Madrid/g
```

De nuevo la expresión devuelve el número de sustituciones hechas, que será 0 (falso) o un número mayor que 0 (verdadero).

Si nosotros queremos sustituir también las ocurrencias de "mAdrid", "madRID", "MaDRiD" y demás, entonces podríamos usar

```
s/[Ma] [Aa] [Dd] [Rr] [Ii] [Dd] /Madrid/g
```

Pero una manera más fácil es usar la opción **'i'** (para ignorar diferencia entre mayúsculas-minúsculas). La expresión

```
s/madrid/Madrid/gi
```

nos haría una sustitución global ignorando mayúsculas-minúsculas. La opción **'i'** es usada también en la coincidencia de las expresiones regulares básicas.

Recordando Patrones

A menudo es útil recordar patrones de concidencias para usarlos de nuevo. Esto lo podemos hacer simplemente anidándolos entre paréntesis, ya que serán guardados en las variables **$1,...,$9**. Estas cadenas pueden ser usadas también en la misma expresión regular (o sustitución) mediante los códigos especiales **\1,...,\9**. Por ejemplo,

```
$_ = "Esto Es una Prueba";
s/([A-Z])/:\1:/g;
print "$_\n";
```

sustituirá cada letra en mayúsculas por la misma letra rodeada por dos puntos. La salida por pantalla será ":E:sto :E:s una :P:rueba". Las variables **$1,...,$9** son variables de sólo-lectura, no se pueden alterar directamente.

Otro ejemplo sería

```
if (/(\b.+\b) \1/)
{
      print"Encontradas $1 repeticiones\n";
}
```

que identificará cualquier palabra que se encuentre repetida. Cada **'\b'** representa una palabra de cada extremo y **'.+'** cualquier cadena no vacía, así que **'\b.+\b'** encaja con cualquier cosa entre dos palabras límite. Esto es recordado por los paréntesis y almacenado como **'\1'** para las expresiones regulares y como **'$1'** para el resto del programa.

Lo siguiente intercambia los primeros y últimos caracteres de una línea en la variable **$_**

```
s/^(.)(.*)(.)$/\3\2\1/
```

Los símbolos **'^'** y **'$'** marcan el principio y final de línea. El código **'\1'** almacena el primer carácter; el código **'\2'** todo lo que esté situado delante del último carácter, el cual es almacenado a su vez en **'\3'**. Luego la línea entera es sustituida con **'\1'** y **'\3'** intercambiados entre sí.

Después de una coincidencia, se pueden usar las variables especiales de sólo-lectura **$`, $& y $´** para averiguar el orden de coincidencias, durante y después de la búsqueda. De manera que después de

```
$_ = "Esto Es una Prueba";
/n/;
```

todo lo siguiente es cierto. (Recordemos que **'eq'** simboliza la equivalencia de cadenas)

```
$` eq "Esto Es u";
$& eq "n";
$´ eq "a Prueba";
```

La función **tr** permite realizar traducciones carácter por carácter. La siguiente expresión sustituye cada **'a'** con una **'e'**, cada **'b'** con una **'d'**, y cada **'c'** con una **'f'** en la variable **$frase**. La expresión devuelve el número de sustituciones hechas.

```
$frase =~ tr/abc/def/
```

La mayoría de los códigos especiales de las expresiones regulares no se aplican a la función **tr**. Por ejemplo, la sentencia siguiente cuenta el número de

asteriscos en la variable **$frase** y almacena la cifra en la variable **$cuenta.**

```
$cuenta = ($frase =~ tr/*/*/);
```

Como siempre, el guión es usado para indicar "entre". Veamos una orden para convertir el contenido de la variable **$_** en mayúsculas

```
tr/a-z/A-Z/;
```

Dispersión de una cadena

Una de las funcionalidades más útiles de Perl es la de dispersar una cadena y recolocarla en un array. La función que se encarga de esto, hace también uso de las expresiones regulares y se aplica sobre la variable especial **$_** a menos que se especifique lo contrario.

La función de dispersión se usa así:

```
$info = "Isasi:Eva:Dentista:20, Torrontes";
@personal = split(/:/, $info);
```

que haría el mismo efecto que

```
@personal = ("Isasi", "Eva", "Dentista", "20,
    Torrontes");
```

Si tenemos almacenada la información en la variable **$_** entonces podemos usar simplemente la sentencia

```
@personal = split(/:/);
```

Si los campos están divididos por un número variable de dos puntos, haremos uso de los códigos de las expresiones regulares.

```
$_ = "Gibson:Mel::Actor:::Sunset Street";
@personal = split(/:+/);
```

o también

```
@personal = ("Gibson", "Mel", "Actor", "Sunset
    Street");
```

En cambio, introducir la siguiente línea

```
$_ = "Gibson:Mel::Actor:::Sunset Street";
@personal = split(/:/);
```

equivaldría a

```
@personal = ("Gibson", "Mel", "", "Actor", "", "",
    "Sunset Street");
```

Una palabra puede ser dispersada en caracteres simples, una frase en palabras y un párrafo en frases.

```
@caracteres = split(//, $palabra);
@palabras = split(/ /, $frase);
@frases = split(/\./, $parrafo);
```

Arrays Asociativos

Los arrays más comunes nos permiten acceder a sus elementos mediante un índice, normalmente suele ser un entero que indica qué elemento del array queremos acceder (primero, segundo, ...). El primer elemento del array @comida es $comida[0]. El segundo es $comida[1], y el resto ya nos los imaginamos. Pero Perl nos permite también crear arrays que son accedidos mediante una cadena, estos son los llamados *arrays asociativos*.

Para definir más profundamente lo que es un *array asociativo* usaremos las notaciones usuales basadas en paréntesis, pero anteponiendo al propio array el símbolo de porcentaje **'%'**.

Supongamos que queremos crear un array de personas y sus edades. Sería algo así como:

```
%edades = ("Eva Isasi", 20, "Mel Gibson", 41,
    "Maripili", "muchos tacos");
```

Ahora podemos averiguar la edad de las personas mediante las siguientes expresiones

```
$edades{"Mel Gibson"}; # Devuelve 41
$edades{"Eva Isasi"};  # Devuelve 20
$edades{"Maripili"};   # Devuelve "muchos
    tacos"
```

Démonos cuenta que al igual que los arrays de listas, cada símbolo **'%'** ha sido cambiado a **'$'** para acceder a un elemento en concreto, ya que ese elemento no es sino un escalar. Contrariamente a los arrays de listas, el índice (en este caso el nombre de la persona) está rodeado por llaves.

Un array asociativo puede ser convertido en un array de lista simplemente asignándolo a una variable array de lista y viceversa para convertir un array de lista en un array asociativo. Luego sólo tendremos que modificar la manera en la que accederemos a la información de los elementos del array, puesto que el índice ya no será el mismo:

```
@info = %edades;          # @info es un array de
                          lista. Ahora
                          # tiene 6 elementos.
$info[3];                 # Devuelve el valor 41 del
                          array
                          # de lista @info
%masedades = @info;       # %masedades es un array
                          asociativo.
                          # Es lo mismo que %edades
```

Los arrays asociativos no tienen un orden predeterminado de sus elementos (son como tablas hash) pero es posible acceder a sus elementos usando las palabras reservadas **'keys'** y **'values'** (clave del array y su correspondiente valor).

```
foreach $persona (keys %edades)
{
    print "Yo sé la edad de $persona\n";
}
foreach $edad (values %edades)
{
    print"Tiene $edad\n";
}
```

Cuando **'keys'** es usado, devuelve una lista de las claves (índices) del array asociativo. De manera similar se hace llamando a **'values'** (valores asociados a las claves o **'keys'**) que nos devolverá la

lista de valores del array. Estas funciones devuelven las listas en el mismo orden, que nada tiene que ver con el orden en el que han sido insertados los elementos en el array.

Cuando **'keys'** y **'values'** son llamados en el contexto de un escalar devuelven el número de parejas (key/value) del array asociativo.

Existe también una función **'each', esta** función devuelve una lista de dos elementos con la clave y su valor asociado.

Cada vez que **'each'** es llamado obtendremos otro par clave-valor (key/value).

```
while (($persona, $edad) = each(%edades))
{
        print "$persona tiene $edad\n";
}
```

Variables de entorno

Cuando se ejecuta un programa Perl o cualquier script en UNIX, hay ciertas variables de entorno ya definidas tales como USER, que contiene el nombre del usuario y DISPLAY que especifica cuál va a ser la pantalla donde se van a mostrar los gráficos.

Cuando se lanza un CGI Perl en una página HTML, también existen algunas variables de entorno que guardan otra información de utilidad. Todas estas variables y sus valores son guardados en el array asociativo %ENV en los que las claves son los nombres de las variables.

Así, podríamos intentar con éxito utilizar el siguiente código en un programa Perl:

```
print "Te llamas $ENV{'USER'} y estás ";
print "usando el monitor $ENV{'DISPLAY'}\n";
```

Subrutinas o Procedimientos (funciones)

Como todo buen lenguaje de programación, Perl permite al usuario definir sus propias funciones, llamadas subrutinas. Pueden ser situadas en cualquier lugar del programa pero probablemente lo mejor sea ponerlas todas juntas al principio o al final del código. Una subrutina tiene la forma

```
sub misubrutina
{
        print "No es una subrutina muy interesan-
        te\n";
        print "Hace siempre lo mismo\n";
}
```

La anterior subrutina hace siempre lo mismo independientemente de los parámetros que la pasemos en su llamada. Para llamar a un procedimiento, usaremos pues el símbolo '&' al principio del nombre:

```
&misubrutina;
&misubrutina($_);
&misubrutina(1+2, $_);
```

En base al código de la anterior función, deducimos que las tres llamadas anteriores (con cero, uno y dos parámetros) nos devolverán el mismo resultado.

Los parámetros son leídos pero ignorados puesto que no se usan en ninguna parte del interior de la subrutina.

Cuando el procedimiento es llamado, los parámetros son pasados como una lista en la variable array especial @_. Esta variable no tiene absolutamente nada que ver con la variable escalar **$_**.

La siguiente subrutina saca por pantalla la lista que se le mande como argumento.

```
sub contenido
{
        print"@_\n";
}

&contenido("rey", "princesa"); # Muestra "rey
        princesa"
```

Como cualquier otro array de lista, los elementos individuales de '@_' pueden ser accedidos directamente usando corchetes:

```
sub primerosegundo
{
        print"El primer argumento fue $_[0]\n";
        print"y $_[1] fue el segundo\n";
}
```

Recordemos de nuevo que los escalares con índice **$_[0]**, **$_[1]** y demás, nada tienen que ver con el escalar **$_** que también puede ser usado sin ningún miedo.

El resultado de una subrutina es siempre la última cosa evaluada. Esta subrutina devuelve el máximo de dos parámetros de entrada.

```
sub maximo
{

        if ($_[0] > $_[1])
        {
         $_[0];
        }
        else
        {
         $_[1];
        }
}

$mayor = &maximo(30, 35);   # Ahora $mayor es
        35
```

La subrutina **&primerosegundo** también devuelve un valor, en este caso 1. Esto sucede porque la última cosa que hizo dicha función fue procesar la sentencia **'print'** y el resultado de esta operación cuando se efectúa con éxito es siempre 1.

Variables Locales

La variable @_ es local al actual procedimiento, al igual que lo son **$_[0]**, **$_[1]**,

$_[2], etc... Se pueden convertir otras variables en locales, lo cual es especialmente útil si queremos empezar alterando los parámetros de entrada. La siguiente subrutina mira si una cadena se encuentra en el interior de otra, los espacios no se toman en cuenta.

```
sub dentro
{
        local($a, $b)
        ($a, $b) = ($_[0], $_[1]);
        $a =~ s/ //g;
        $b =~ s/ //g;
        ($a =~ /$b/ || $b =~ /$a/);
}

&dentro("tomate", "plato mate");        #
        Devuelve verdadero
```

De hecho, podemos compactar también las dos primeras líneas de la subrutina con ésta:

```
local($a, $b) = ($_[0], $_[1]);
```

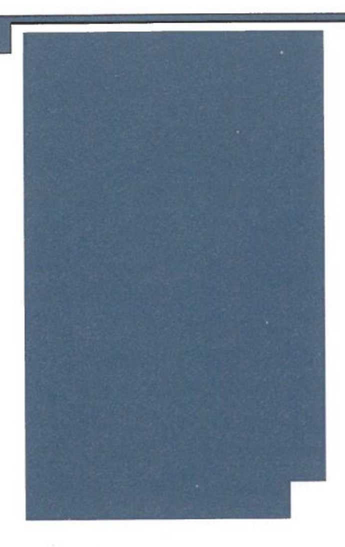
Conclusión

Pues bien, con esto terminamos este pequeño Curso de Programación en Perl. Por supuesto la cosa no queda ahí, se pueden realizar programas algo más complejos con todo lo enseñado y un buen manual de referencia al lado.

El campo de aplicación de este lenguaje, como ya dijimos, no es otro que el tratamiento de ficheros de texto o formularios. Podemos crear preciosas aplicaciones de análisis de 'logs', tratamiento y proceso de formularios de páginas HTML, etc...

Perl no es un lenguaje que haya sido creado siguiendo unas estrictas normas de sintaxis; su uso se limita a la creación de aplicaciones lo más rápidamente posible, con lo que se sacrifica cierta modularidad y orden. Los programas Perl se crean para dar solución a un problema en concreto y ya no tener que volver a mirar el código de cómo lo hemos resuelto nunca más. Ha sido diseñado para salir del paso de tareas bastante concretas, con lo que no lo tachéis de incompleto, incomprensible o poco modular.

Como ya os advertimos, Perl es un lenguaje muy sencillo en lo que se refiere a su programación, aunque eso sí, bastante ilegible. Podéis hacer la prueba, y es que si no documentamos bien el código fuente que creemos, al cabo de un par de días que tengamos que extraerlo para revisarlo y/o modificarlo, nos encontraremos con un amasijo de símbolos extraños imposibles de descifrar. Así pues estáis advertidos, si pensáis mantener el programa periódicamente o con sucesivas versiones, tened cuidado a la hora de implementar expresiones regulares creadas por vosotros mismos y describidlas detalladamente como si el código tuviera que ser manipulado por otros programadores.



Correo del lector

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relacionado con la programación.

Pregunta

Hola. Soy un lector de la revista que acaba de instalar Windows NT 4.0 Workstation como servidor de una red de tres puestos que se ejecutan bajo Windows95. Hace un par de días instalé unos programas en dicho servidor, procediendo luego a su desinstalación. Creo que he hecho algo mal puesto que a la hora de desinstalarse me preguntó si quería eliminar unas librerías las cuales podrían ser utilizadas por otros programas. Elegí la opción de aceptar y ahora me encuentro con que el sistema operativo no me arranca si quiera, dándome un error servero en el fichero **ntoskrnl.exe**.

El mensaje que me aparece por pantalla me aconseja arrancar con otra configuración Hardware, pero yo sólo definí una.

Me gustaría saber cómo puedo recuperar el sistema sin tener que reinstalarlo de nuevo y configurar otra vez desde cero los programas.

Respuesta

Bien, la pregunta que nos haces tiene una muy sencilla solución. Al contrarío que Windows95, Windows NT4 posee mayores facilidades a la hora de reparar desperfectos producidos en el arranque del sistema, los ficheros de registro o incluso los discos duros, ya sean con sistema de ficheros VFAT ó NTFS.

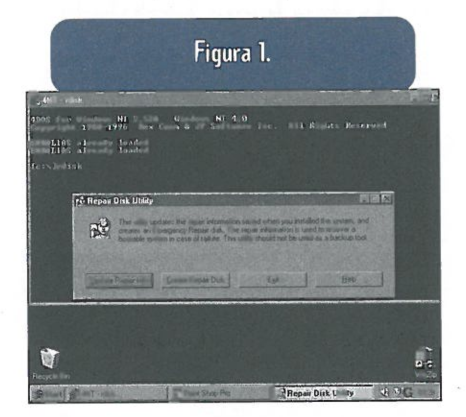
El sistema operativo Windows NT4, tanto en su versión Wokstation como en la versión Server, posee una excelente protección contra fallos o cuelgues de programas, portándose muy bien a la hora de cerrar aplicaciones conflictivas.

Tu error lo da nada más arrancar así que efectivamente es un error grave puesto que no ha dado tiempo al sistema ha cargar siquiera el anillo básico o kernel.

Para saber más del error que te da, teclea la opción **/CRASHDEBUG** en el fichero editable **boot.ini**.

Para solucionar este tipo de errores u otros que impidan el arranque del sistema aconsejamos hacer uso de los tres discos que NT creó para su instalación. Una vez que ha arrancado el pequeño kernel que se encuentra en éstos, detectará la existencia de una versión de NT4 ya instalada y ofrecerá una opción adicional de reparar el sistema.

Figura 1.



Esta opción permitirá instalar de nuevo todo el núcleo del sistema incluido su arranque en el disco, manteniendo la configuración de todos los dispositivos y programas.

Conviene que a la hora de especificar qué partes recuperar, no seleccionemos los ficheros de registro, puesto que perderíamos toda la configuración de los programas instalados en el sistema, enlaces de extensiones de objetos (.JPG con el Paint Shop Pro...), etc... Esto sólo lo haremos en caso de una instalación defectuosa o incompleta de algún programa, y siempre y cuando hayamos creado recientemente un disco de recuperación.

Aunque la opción de recuperar funciona también sin el mencionado disco, recomendamos crear éste habitualmente, sobre todo cuando hayamos modificado el sistema o hayamos instalado algún programa para poseer el mayor grado de seguridad. Para crear el disco de emergencia, teclearemos el comando **RDISK** desde una sesión MSDOS del NT y seguiremos las instrucciones de pantalla, ya sea crear un disco nuevo o actualizar uno ya existente.

DB2 Universal DataBase, GetRight 3.02, Microsoft Project 98 Trial

CONTENIDO DEL CD

Este mes, el CDROM está dedicado muy especialmente a todos aquellos que nos han solicitado incluir un mirror de algún site de Internet de dominio público.

Aprovechando la finalización del "Curso de Programación en Perl", hemos extraído el Comprehensive Perl Archive Network (CPAN), famoso por ser un site de lo más completo de Perl del que existen multitud de mirrors debido a la gran cantidad de visitas que recibe. En estos casi 300MB podrás encontrar todo lo conocido sobre Perl además de muchísimas utilidades y la propia distribución, acompañada de un montón de FAQs sobre este lenguaje de programación. Todo lo que esté escrito en Perl y sea útil o de libre distribución estará en el CPAN.

La arquitectura del CPAN es una creación de Jarkko Hietaniemi y Andreas König. Andreas y Tim Bunce mantienen en común la lista de módulos para Perl 5 e informan de la creación de estos módulos más que de lo que hacen. Los paquetes están ordenados por autor, lo cual complica de alguna manera saber que hacen exactamente, aunque se incluyen páginas Web cumpliendo con esta tarea.

Estos módulos están completamente documentados y versionados y comprenden toda clase de utilidades a todos los niveles, desde utilidades CGI a utilidades capaces de hacer mirrors de sites en la red de área global. Podremos encontrar el mirror del CPAN en el directorio con el mismo nombre situado en el raíz: \CPAN, lo que estamos seguros que ahorrará mucho dinero en facturas telefónicas a nuestros lectores.

En esta misma sección de programación, podéis encontrar también la versión de evaluación del Borland DataGateway for Java, aplicación de Borland que proporciona a los desarrolladores bajo JDBC un entorno fiable al mismo tiempo que rápido como solución para la conectividad entre bases de datos aportando facilidades concurrentes multi-hilo.

Linux

Los usuarios de Linux encontrarán uno de los paquetes más famosos de este sistema operativo, Star Office 4.0. Se trata de una suite al estilo del Office de Microsoft que proporciona un conjunto de herramientas que se integran perfectamente entre sí y que incorporan además funciones avanzadas de correo electrónico. Se trata de una versión Beta limitada en uso hasta el 31 de Enero de 1998.

También incluimos, debido a la voz de alarma despertada en Internet, un parche que deberían instalar todos aquellos administradores que estén manteniendo un servidor bajo este sistema operativo, a fin de evitar ataques por parte de hackers con malas intenciones (que también los hay).

Redes

La familia DB2 de productos sobre bases de datos relacionales, ofrece manejo de bases de datos de calidad industrial para toma de decisiones, proceso de transacciones, y un extensivo rango de aplicaciones de negocios. La familia DB2 se extiende a sistemas AS/400, hardware de tipo RISC System/6000, Mainframes IBM, máquinas no-IBM del tipo Hewlett-Packard y Sun Microsystems, y sistemas operativos como OS/2, Windows (95&NT), AIX, HP-UX, SINIX, SCO OpenServer, y entorno operativo Solaris.

Teniendo en cuenta la versatilidad de estas aplicaciones teníamos dudas de incluirlas entre la sección de programación o la sección de Redes, ya que más que un producto enfocado a la programación, está abocado a la interconexión de Bases de datos relacionales.

El producto ofrece 60 días de prueba desde su instalación. La versión aquí presentada es la de IBM OS/2 Warp tanto ser-

vidor como clientes. En próximas entregas se facilitarán versiones para Windows 95 y NT, que no se han incluido este mes por falta de espacio. Además IBM proporciona una completa documentación navegable desde el CD ya que está en formato HTML.

DB2 Universal Database.

IBM SEARCH Software

SPECIAL ANNOUNCEMENTS



DB2 Product Family

IBM Dominates 1997 Database Awards: Ranked #1 in DBPD Database Dozen, Sweeps nine of 10 categories in VARBusiness ARC Awards. See the new Data Awards page for details.

Download or free CD-ROM: get a 60-day trial version of DB2 Universal Database 5.0 for Windows NT/95, OS/2

New! DB2 Universal Database 5.0 for AIX, HP-UX, and Sun Solaris: Request a free CD-ROM with NFR code



Varios

En la medida de lo posible vamos a intentar incluir habitualmente en esta sección aquellos parches o actualizaciones de herramientas comerciales de más uso que permiten mantenerse a la última sin apenas esfuerzo, ya que muchas veces, debido al gran tamaño que ocupan estos ficheros en Internet, a más de uno le es imposible obtenerlos exceptuando algunos pocos privilegiados poseedores de una RDSI.

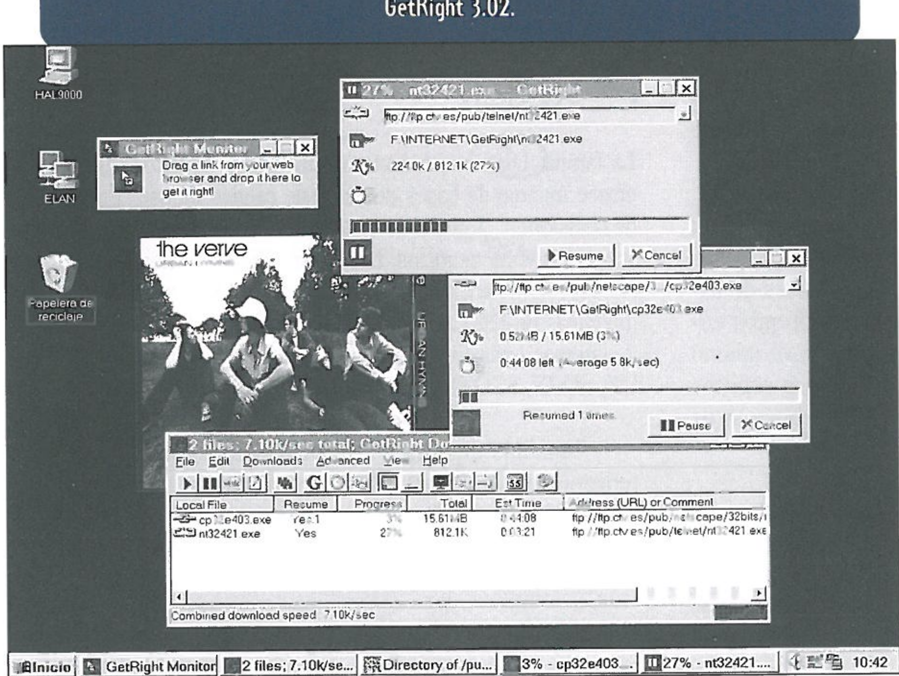
Este mes hallaréis aquí el Service Pack 3 que actualiza la herramienta Visual Basic 5.0 de algunos bugs no corregidos en la última entrega.

Asimismo, como ya lo hiciera antes con FrontPage, Microsoft ya ha empezado el ataque de versiones 98 de sus más programas más famosos. Esta vez suministramos a modo de evaluación y con un período limitado a 60 días a partir de su instalación, la herramienta Microsoft Project 98 para Windows 95 y NT4.

Resumen CD-ROM.

- Mirror del (CPAN) Comprehensive Perl Archive Network
- DB2 Universal Database
- Borland DataGateway for Java
- Star Office 4.0 para Linux
- GetRight 3.02
- Service Pack 3 del Visual Basic 5.0
- Microsoft Project 98 Trial

GetRight 3.02.



Peticiones

Recordamos como cada mes a todos los lectores a que participen de manera activa en esta sección y envíen peticiones de programas, código fuente, etc... para incluir en el CDROM, siempre y cuando éstos sean de libre distribución o tengan una licencia restringida a un período de tiempo limitado a modo de evaluación. Así pues mostramos una vez más nuestra dirección de correo electrónico destinada a tal fin: solop@towercom.es. Utilizadla tanto como deseéis, destacando como <tema> del mensaje: "Sugerencias CDROM".

No hacemos servidores, simplemente los mejoramos

En Adaptec dejamos los servidores para los expertos. Pero cuando sus clientes necesitan más velocidad, más seguridad y más conectividad, acuden a nosotros.

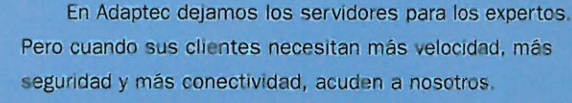
SCSI - Los adaptadores SCSI de Adaptec le permiten conectar hasta 15 periféricos a su servidor y son capaces de doblar la velocidad a la que puede mover los archivos por el sistema.

RAID - La serie de adaptadores de matriz AAA-130 de Adaptec hace de RAID una opción viable para servidores con Windows NT y NetWare. Considerado como el concepto más fiable de almacenamiento de datos, RAID divide los datos y los coloca en varias unidades, con lo que le proporciona un acceso más rápido y seguro.

FAST ETHERNET - Si desea conectar su servidor a una red Fast Ethernet, el adaptador de 4 puertos Quartet de Adaptec cuadruplica su número de conexiones y sostiene velocidades de hasta 100 Mb/s. Quartet también es fácil de administrar con la suite de software Duralink.

FIBRE CHANNEL - Para almacenamiento externo y grupos de servidores, los controladores Fibre Channel de Adaptec aseguran un rendimiento inmejorable. Fibre Channel es la última tecnología para subsistemas de almacenamiento en servidor y proporciona la friolera de 100 Mb/s en distancias de hasta 10 Km para hasta 126 dispositivos.

Si su servidor es simplemente estándar, su rendimiento también lo será. Póngase en contacto con Adaptec, el líder mundial en tecnología de transferencia de datos, y juntos haremos que su servidor sea superior.



http://www.adaptec.com.

